T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

BÜYÜK VERİ ÜZERİNDE DUYGU ANALİZİ YÖNTEMLERİ VE AZERBAYCAN DİLİNE UYGULANMASI

Doktora Tezi

Huseyn HASANLI

Matematik Anabilim Dalı

İzmir
2019

T.C.
EGE ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü

BÜYÜK VERİ ÜZERİNDE DUYGU ANALİZİ YÖNTEMLERİ VE AZERBAYCAN DİLİNE UYGULANMASI

Huseyn HASANLI

Danışman: Doç. Dr. Burak ORDİN

Matematik Anabilim Dalı

Bilgisayar Bilimleri Doktora Programı

İzmir
2019

Huseyn **HASANLI** tarafından doktora tezi olarak sunulan "Büyük Veri Üzerinde Duygu Analizi Yöntemleri Ve Azerbaycan Diline Uygulanması" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 20.12.2019 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri:		**İmza**
Jüri Başkanı	: Doç. Dr. Burak ORDİN	.....
Raportör Üye	: Dr. Öğr. Üyesi Arif GÜRSOY	.....
Üye	: Prof. Dr. Urfat NURİYEV	.....
Üye	: Prof. Dr. Efendi Nasiboğlu	.....
Üye	: Dr. Öğr. Üyesi Onur UĞURLU	.....

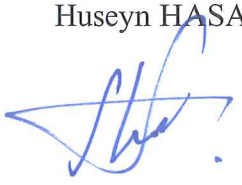
EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Doktora Tezi olarak sunduğum "Büyük Veri Üzerinde Duygu Analizi Yöntemleri Ve Azerbaycan Diline Uygulanması" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim

20 / 12 / 2019

Huseyn HASANLI



ÖZET

BÜYÜK VERİ ÜZERİNDE DUYGU ANALİZİ YÖNTEMLERİ VE AZERBAYCAN DİLİNE UYGULANMASI

HASANLI, Huseyn

Doktora Tezi, Matematik Anabilim Dalı
Tez Danışmanı: Doç. Dr. Burak ORDİN
Aralık 2019, 68 sayfa

Son zamanlarda mikro blok siteleri internet kullanıcıları arasında çok popüler bir iletişim aracı haline gelmiştir. Milyonlarca kullanıcı her gün kendi yaşamının farklı yönleriyle ilgili görüşlerini, duygularını farklı bloklarda paylaşmaktadır. Bu nedenle mikro blok siteleri, fikir madenciliği ve duyarlılık analizi için zengin veri kaynaklarına çevrilmektedir.

Duygu Analizi (DA), metin madenciliği alanında devam eden bir araştırma alanıdır. Duygu Analizinin görevi, kişilerin görüşlerini verilen bir metinden pozitif, negatif veya nötr gibi farklı kategoriler olarak etiketlemektir.

Bu çalışmada duygu analizi ve fikir madenciliği için oluşturulan Azerbaycan dilinde tweet verisinin toplama, temizleme ve etiket ekleme ilkeleri açıklanmaktadır. Ayrıca maksimum entropi, naive bayes, destek vektör makinesi gibi makine öğrenme yöntemler ve sözlük tabanlı kelime sayım ve özellik puanlama yöntemleri kullanılmıştır. Elde edilen sonuçlar ve karşılaştırılmış ve tweetlerin sınıflandırılması için optimal parametreler tanımlanmıştır.

Literatürde yapılan çalışmaların büyük bölümünde kullanılan veriler İngilizce içeriklidir. Bundan dolayı Azerbaycan dili için duygu analizi alanında daha fazla kaynak sunabilmek adına, bu konuda doktora tezinin yapılmasına karar verilmiştir. Bu çalışmada Azerbaycan dilinde duygu analizi yapabilmek için hibrid bir yaklaşım içeren bir sistem geliştirilmiştir. Çalışmada kullanılan dil Azerbaycan dili olsa da, önerilen teknik diğer dillerle de kullanılabilir.

Anahtar sözcükler: Duygu analizi, Sınıflandırma, Twitter, Makine öğrenme, Azerbaycan dili

ABSTRACT

SENTIMENT ANALYSIS METHODS ON BIG DATA AND APPLICATION TO AZERBAIJAN LANGUAGE

HASANLI, Huseyn

Doctoral Thesis, Department of Mathematics
Supervisor: Assoc. Prof. Burak ORDIN
December 2019, 68 pages

In recent days microblog sites have become a very popular means of communication among internet users. Millions of users share their opinions and feelings about different aspects of their lives in different blocks every day. For this reason, microblog sites are translated into rich data sources for idea mining and sentiment analysis.

Sentiment Analysis (SA) is an ongoing research area in the field of text mining. The task of Sentiment Analysis is to label people's views from a given text into different categories, such as positive, negative or neutral.

In addition, machine learning methods such as maximum entropy, naive bayes, support vector machine and lexicon based word counting and feature scoring methods were used. The results obtained were compared and the optimal parameters for the classification of tweets were defined.

The data used in most of the studies are in English. Therefore, in order to provide more resources in the field of emotion analysis for the Azerbaijani language, it was decided to do a doctoral thesis on this subject. In this study, a hybrid approach was developed to analyze sentiment in the Azerbaijani language. Although the language used in the study is with the Azerbaijani language, the proposed technique can be used with other languages.

Keywords: Sentiment Analysis, Classification, Machine Learning, Twitter, Azerbaijani language.

ÖNSÖZ

Duygu ve düşünce, insan varlığının temel özellikleridir. "Ne düşünüyoruz ?" ve "nasıl hissediyoruz ?" günlük yaşamımızda hayati bir rol oynamaktadır. Aldığımız kararlar hem kendimizin hem de başkalarının duygu ve düşünceleriyle yakından ilgilidir.

Duygu analizi araştırması, son yıllarda, kullanıcının bireysel düşüncelerini ve fikirlerini yayınlayabileceği sosyal medya araçlarının yaygınlaşması ile birlikte ivme kazandı. Sosyal medyanın popülaritesinin artması, dünya çapındaki ağı statik bir depodan, herkesin dünya genelinde görüşlerini dile getirmesi için dinamik bir foruma dönüştürmüştür.

Doktora tez konumu belirlerken günümüzde popüler olan sosyal medya verilerini analiz etmek istedim ve tezimi bu doğrultuda “Büyük Veri Üzerinde Duygu Analizi Yöntemleri Ve Azerbaycan Diline Uygulanması” olarak belirledim. Büyük veri kaynağı olarak ise Azerbaycan dilinde paylaşılan Twitter verilerinin kullanılmasına karar verdim. Ayrıca problem üzerinde dünya çapında farklı diller için de çok yaygın çalışılmadığını görmem, konuya olan ilgimi arttırmıştır. Azerbaycan dili için yapılan çalışmalara önemli kazanç sağlamağı kendime hedef belirledim.

Bu tezin başlangıcından tamamlanıncaya kadar karşılaştığım birçok zorluklara değerli hocalarımın ve çevremdekilerin desteğiyle üstesinden geldim ve Azerbaycan dili için yapılan akademik kaynaklara katkı sağlayacağına inandığım bu tezi hazırladım.

İZMİR

20/12/2019

Huseyn HASANLI

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
e_j	Hedef varlık
a_{jk}	Varlığın bir özelliği /görünüşü
so_{ijkl}	Duyarlılık değeri
h_i	Fikir veya görüş sahibi
t_1	Görüşün ifade edildiği zaman
i	İşlemin başlangıç düğümü

Kısaltmalar

DA	Duygu Analizi
FM	Fikir Madenciliği
SVM	Support Vector Machine
NB	Naïve Bayes
YSA	Yapay Sinir Ağları
ME	Maksimum Entropi
TF	Terim Frekansı
TDF	Ters Doküman Frekansı
MÖ	Makine Öğrenme

1. GİRİŞ

Geçtiğimiz birkaç yıldan itibaren, sosyal medya modern yaşamda hayati bir rol oynamaktadır. Sosyal medya kullanıcılarının sayısı günden güne artmaya devam ediyor. Kullanıcılar bu platformda görüşlerini, düşüncelerini, yaşam olaylarını hiçbir kısıtlama ve tereddüt etmeden yayınlayabiliyorlar. Sosyal medyadaki mesaj biçimindeki milyarlarca metin verisi araştırmacıların veri analizi yapabilmesi için çok etkileyici bir araç olmaktadır. Sosyal medya portalları, genel olarak metin tabanlı mesajlar ve resimler yoluyla duyguları ifade etmek için için kullanılmaktadır (Jain and Katkar, 2015). Şu anda, Twitter, Facebook, LinkedIn, Flickr vb portallar kullanıcıların görüşlerini herkese açık olarak yayınlamalarına izin vermektedir. İnsanlar etkileşimde bulunmak ve yorumları paylaşmak için çeşitli sosyal medya platformlarını kullanabilse de, bu tez kapsamında kullanılacak kaynak olarak Twitter seçilmiştir. Tüm sosyal medya araçları arasında Twitter, belirli bir konuyla ilgili insanların görüşlerini ifade etmek için yaygın olarak kullanılmaktadır.

Gerçek hayata bakıldığında insanların farklı konulara karşı farklı fikirleri bulunmaktadır. Kişilerin herhangi bir konu hakkında fikirlerini öğrenebilmek için her ne kadar anket vb. yöntemler bulunsa da, günümüzde her bir bireyin etrafında bulunan ve büyük hızla artan veri olması nedeniyle bu yöntemler çok da pratik değildir. İnternet ve beraberinde sosyal paylaşım platformlarının yaygınlaşmasıyla bu platformlar üzerinde sanal bir yaşam oluşmuştur ve bu yaşam üzerinde insanlar fikirlerini ifade etmek, çok geniş yelpazede farklı konuları tartışmak, yeni fikirlere sahip olmak vb. anlayışlara sahip olmaktadırlar. Günümüzde özellikle Twitter platformu sosyal medyada insanların düşüncelerini ifade etmede yaygın olarak kullanılmaktadır. Twitter'dan toplanan veriler ile doğal dil işleme ve veri madenciliği ile pek çok araştırma yapılmakta ve endüstride projeler gerçekleştirilmektedir. Sonuç olarak kullanıcılar Twitter'da görüş ve duygularını paylaşarak büyük miktarda veri üretmektedirler. Bu tweet`ler müşterilerin hizmetlerini iyileştirmelerine ve kaliteli ürünler üretmelerine yardımcı olmak için şirketler ve araştırmacılar tarafından incelenmektedir.

Twitter verilerinde duygu probleminin çözümü için çeşitli algoritmalar önerilmiş ve bu alanda farklı çalışmalar yapılmıştır.

Pang ve arkadaşları 2002 yılında yayınladıkları makalede film incelemelerinin duyarlılık sınıflandırması konu temelli metin kategorizasyon probleminin özel bir örneği olarak ele alınmıştır. Üç sınıflandırma algoritması olan : Naif Bayes, Maksimum Entropi ve Destek Vektör Makineleri bu probleme uygulanmış ve sonuçları karşılaştırılmıştır: (Pang et al., 2002)

2004 yılında Minqing Hu ve Bing Liu tarafından yayınlanan makalede internet üzerinden satılan ürünlerin müşteri incelemelerine ait özellik tabanlı özetler çıkarma problemi incelenmiştir. Bu problemde özellikler genel olarak ürün özellikleri ve işlevleri anlamına gelir. Makalede problem üç alt problem olacak şekilde belirlenmiştir: (1) müşterilerin görüşlerini ifade ettiği ürünün özelliklerini belirleme; (2) her özellik için, olumlu veya olumsuz görüşler veren inceleme cümleleri belirleme; ve (3) keşfedilen bilgiyi kullanarak bir özet üretmektir (Liu and Hu, 2004).

2006 yılında Shailesh Kumar Yadav tarafından yayınlanan makalede duyarlılık sınıflandırması, sınıflandırma teknikleri ve duyarlılık analizi için hangi araçların mevcut olduğu üzerinde durulmuştur. Bu alanda karmaşık cümlelerin polaritesini keşif etmek, farklı korpuslardan fikir cümlelerinin ve özelliklerinin çıkarılması ve aynı belgeden çoklu görüşlerin çıkarılması gibi zorluklar olduğu belirtilmiştir (Yadav, 2016).

Mahmood ve arkadaşları 2013 yılında yayınladıkları makalede Pakistan'da yapılan 2013 seçiminin galibinin tahminlenmesinde atılan tweetlerin etkisini analiz etmişler. Bu analiz çalışmasında verilerin eğitimi için, Rapid Miner aracı (http://rapidi.com/) üç farklı standart tahmin modeliyle denemek için kullanılmıştır. Bu yöntemler: CHAID karar ağacı, Naive Bayes ve Support Vector Machine (SVM) olacak şekilde seçilmiştir. CHAID (Ki-kare Otomatik Etkileşim Dedektörü) daha büyük veri setlerinin analizi için çok uygundur (Mahmood et al., 2013).

M. Saravan ve arkadaşları kullanıcılar tarafından gönderilen tweet`leri bulundukları konumlarla ilişkilendirmek için bir yöntem geliştirdiler (Saravan et al., 2013).

Shital ve Jeevan tarafından 2017 yılında yayınlanan makalede müşterilerin yaklaşımlarını ve görüşlerini öğrenmek için Twitter sosyal medya verileri kullanılmıştır. Bu makalede hizmet sağlayıcı firmaların veya sektörlerin müşterilerinin onların ürünlerine, hizmetlerine veya tekliflerine ilişkin görüşlerini bulmalarında yardımcı olan yaklaşım gösterilmektedir. Bunun için ilk olarak tweet`ler alınıp daha sonra Stanford NLP kullanılarak bu tweet`ler olumlu olumsuz ve nötr olacak şekilde sınıflandırılmıştır (Shital and Jeevan, 2017).

Rane ve Kumar tarafından 2018 yılında yayınlanan makalede 6 büyük ABD Havayoluna ait tweet'lerden oluşan veri seti alınmış ve bu veriler üzerinde çok sınıflı duyarlılık analizi yapılmıştır. Bu çalışma tweet'leri temizlemek için kullanılan ön işleme teknikleri ile başlamış daha sonra ise bu tweet'leri ifade düzeyinde bir analiz yapmak için derin öğrenme kavramı (Doc2vec) kullanarak vektörler olarak gösterilmiştir. Analiz 7 farklı sınıflandırma stratejisi kullanılarak yapılmıştır: Karar Ağacı, Rastgele Orman, SVM, K-En Yakın Komşular, Lojistik Regresyon, Gauss Naif Bayes ve AdaBoost (Rane and Kumar, 2018).

Velioğlu ve arkadaşları tarafından 2018 yılında yayınlanan makalede, Türkçe pozitif, negatif ve tarafsız tweet`ler için emojiler (ifadeler) üzerine duyarlılık analizi problemi ele alınmıştır. Bu makalede kelime torbası (bag of words) ve fastText'e dayalı iki farklı yöntem kullanılmıştır. FastText: kelime ve cümle sınıflandırmalarını öğrenmek için Facebook'un AI Research (FAIR) tarafından geliştirilen açık kaynaklı bir kütüphanedir. Yapılan çalışmada ilk olarak basit ve etkili temel bir yöntem olduğu için kelime torbası yaklaşımı uygulanılmıştır. Bu yöntemde Naif Bayes, Lojistik Regresyon, Destek Vektör Makineleri, Karar Ağaçları gibi sınıflandırıcılar tweet`lere uygulanmıştır. İkinci olarak için fastText yöntemi uygulanmış ve iki model arasında anlamlı bir fark olmadığını gösterilmiştir (Velioğlu et al., 2018).

Chory ve arkadaşları tarafından 2018 yılında yayınlanan makalede, Endonezya'da telekomünikasyon operatörünün veri servisinin kullanımında kamu memnuniyeti ile ilgili duyarlılık analizi ele alınmıştır. Veri servisinin kullanıcı memnuniyeti seviyesine ilişkin duyarlılık sınıflandırma sistemi yapmak için makine öğrenme yöntemlerinden olan SVM ile birlikte TF-IDF ağırlıklandırma, POS Etiketleme ve Negatiflik yönetme gibi yöntemler kullanarak sınıflandırma öncesi doğruluğun iyileştirilmesi sağlanmıştır (Chory et al., 2018).

2016 yılında Joshi ve Tekchandani tarafından yayınlanan makalede Twitter'dan alınan film incelemesi veri setine denetimli sınıflandırıcılar olan SVM (Destek Vektör Makinesi), Naive Bayesian ve Maksimum Entropi yöntemlerini uygulayıp sonuçları karşılaştırılmıştır (Joshi and Tekchandani, 2016).

Bu tez, girişi izleyen 7 bölümden oluşmaktadır.

İkinci bölümde Duygu analizi konusu çalışılmış ve uygulama alanları incelenmiştir.

Üçüncü bölümde; Duygu analiz yöntemleri ele alınıp türleri belirtilmiş ve yöntemler içerisinde yer alan kavramlar örneklerle tanıtılmıştır. Duygu analizi problemleri için kullanılan yöntemler incelenmiştir.

Dördüncü bölümde; veri toplama ve önişleme yöntemleri ele alınıp, bu yöntemler detaylı şekilde çalışılmıştır.

Beşinci bölümde; Azerbaycan twitter verileri üzerinde duygu analizi yapmak için geliştirilen sistem anlatılmıştır.

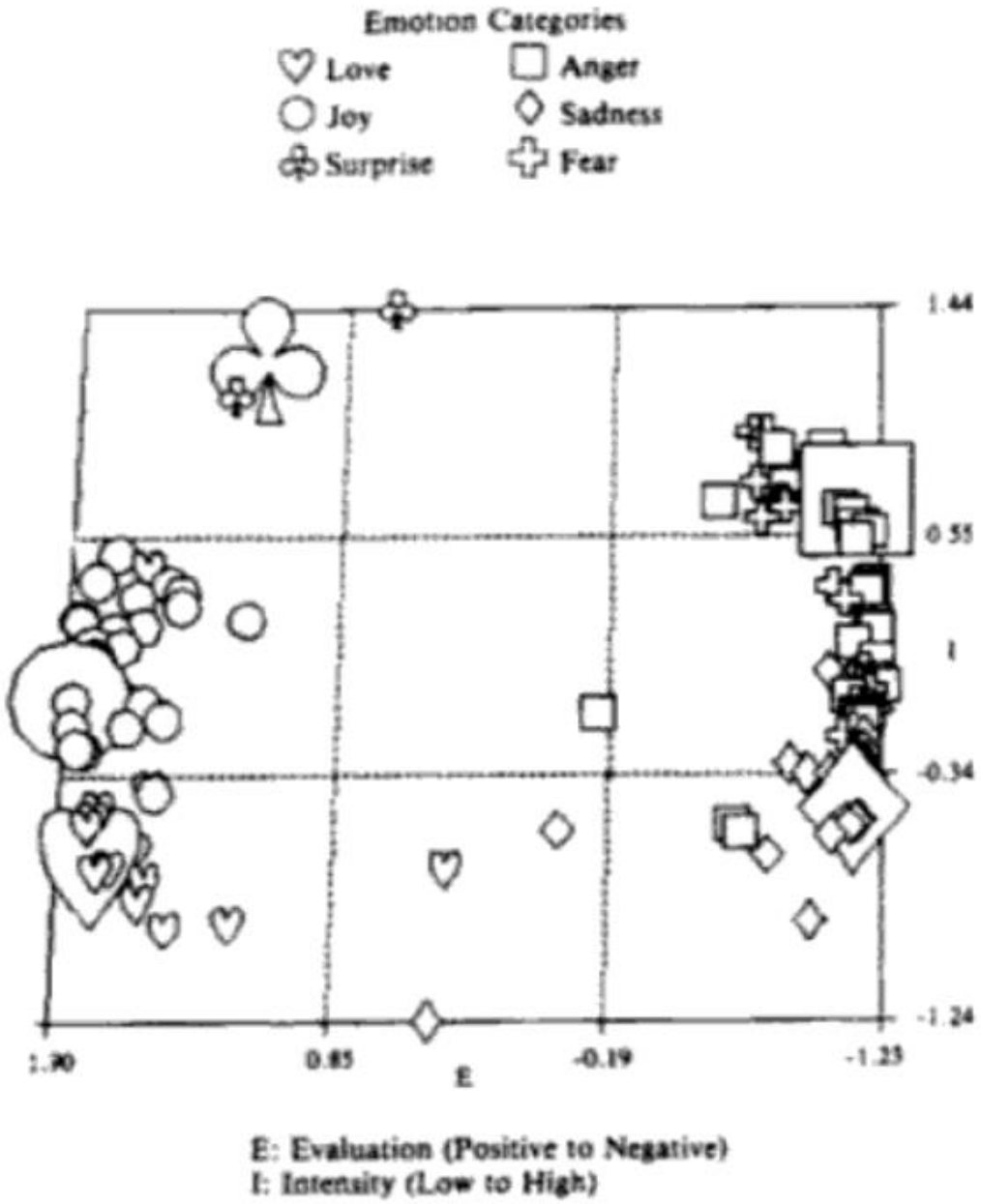
Altıncı bölümde; çözüme yönelik önerilen modelin gerçekleştirimi yapılmış ve elde edilen sonuçlar ortaya konup, tartışılmıştır.

Yedinci bölüm ise sonuç bölümüdür. Bu bölümde yapılan tüm çalışmalar kısaca özetlenmiş ve elde edilen genel sonuçlar sunulmuştur.

2. DUYGU ANALİZİ

Web'in ilk aşamalarında bilgi haber siteleri ve ajanslar gibi geleneksel medya kaynaklarından web sitesi çalışanları tarafından yayınlanıyordu. Bilgi içeriği esas olarak belirli varlıklar veya konular hakkında objektif ifadeler olan "gerçekler" ile ilgiliydi. 2000'lerde bloklar, çevrimiçi sosyal ağlar ve mikro blok hizmetleri gibi Web 2.0 platformlarının (O'Reilly, 2007) yükselişi, kullanıcıların metin içeriğini daha basit bir şekilde oluşturmalarını ve paylaşmalarını sağlayarak durumu değiştirdi. Bu durum web'de sübjektif bilgilerin (yani kişisel görüşlerin) patlamasına neden oldu, bu da bilgi sistemi geliştiricileri için yeni fırsatlar sağlamıştır. Web uygulamalarının ve sosyal medyanın hızlı büyümesiyle kullanıcılar tarafından oluşturulan görüşler, yorumlar, derecelendirmeler ve geri bildirimler ortaya çıkmıştır. Bu görüşler ürünler, politikalar, haberler, kişiler, hizmetler ve etkinlikler dahil neredeyse her şey hakkında olabiliyor. Bunların tümünde kullanıcının ne düşündüğünü ve ne hissettiğini iyi bir şekilde tahmin etmek için işlenmesi ve analiz edilmesi gerekir (Duwairi et al, 2014).

Bu bölümde, araştırma literatüründe genel olarak fikir madenciliği ve duygu analizi teknikleri olarak adlandırılan bu yöntemlerin tanımlanması yapılmıştır. Duygu analizi ve fikir madenciliği, birbirinin yerine geçebilir terimler olacak hale gelmiştir. Duygu analizini anlamak için önce duygu ve duyarlık kavramının kendisinin ne olduğunun tanımlanması gerekmektedir. Duyarlık temelde gerçeklerden ziyade fikir ve tutumlara odaklanırlar; bu yüzden duygular gerçekten özneldir yani kişisel bakış açısı ile ilgilidir. Psikologların altı farklı kategoride organize etmeye çalıştıkları çok sayıda duygu vardır: aşk, sevinç, sürpriz, öfke, üzüntü ve korku. Bunlardan sevinç ve üzüntü gibi duygular her gün farklı derecelerde insanların yaşadığı yaygın duygulardır (Parrott, 2001).



Şekil 2.1. Duygu kategorileri

Duygu analizi metinde bahsi geçen bir varlığa yönelik olarak ifade edilen görüş, duygu, his ve tutumların hesaplamalı bir çalışmasıdır. Duygu analizinin görevi farklı konular veya varlıklar ile ilgili fikir, düşünce ve tutumları tespit etmek, ortaya çıkarmak ve duyarlılık sınıflandırması yapmaktır. Duyarlılık sınıflandırması verilen metnin duyarlılık yönünün iki veya daha fazla sınıfta belirlenmesidir (Ravi, 2015).

Görüşler çoğu zaman insanlar için anlaşılması kolay olsa da bir bilgisayarın aynı anlama seviyesinde anlaması o kadar kolay değildir. Bahsedilen görüş kavramı aşağıdaki maddelerden oluşur:

1) Görüş hedefleri(Opinion targets): varlıklar ve onların özellikler / yönleri
2) Hedef varlığın özelliği /görünüşü
3) Duygular (Sentiments:): olumlu (pozitif), olumsuz (negative) veya nötr (neutral)
4) Görüş sahibi (Opinion holders): görüşleri söyleyen veya yazan kişiler
5) Zaman (Time): görüşler ifade edildiği zaman.

Biçimsel olarak görüş beşli $(e_j, a_{jk}, so_{ijkl}, h_i, t_1)$ olarak temsil edilir. Burada (e_j) hedef varlık, (a_{jk}) varlığın bir özelliği /görünüşü, (so_{ijkl}) duyarlılık değeri, (h_i) fikir veya görüş sahibi ve (t_1) görüşün ifade edildiği zamandır (Appel et al., 2015).

SELMA:"Helal olsun Milli takım EURO 2020 Elemeleri H Grubu'nda Arnavutluk karşısında 1-0 galip gelerek yüzümüzü güldüren A Milli Futbol Takımımızı yürekten kutluyorum Tebrikler 👏👏👏"(12 ekim 2019)

Örnek olarak kullanılan tweet`de : (e_j) hedef varlık *"Milli takım"*, (a_{jk}) varlığın özelliği veya görünüşü *"galip", "güldüren",* (h_i) görüş sahibi "*SELMA*", (t_1) görüşün ifade edildiği zaman *"(12 Ekim 2019)"* ve (so_{ijkl}) duyarlılık değeri ise olumlu kelimelerinin daha fazla olduğunu dikkate alarak *"olumlu"* bir tweet olduğu belirtilebilir.

Duygu analizi, hesaplamalı dilbilim ve veri madenciliği alanındaki alt disiplindir. Duygu analizinin temel amacı insanların ruh hallerini, davranışlarını ve fikirlerini metin verilerinden keşfetmektir. Mikro blok hizmetlerinin artmasıyla birlikte, bu hizmetlerin halka açık verileri; politika, ekonomi ve finans gibi farklı sosyolojik alanlara yönelik veriler duygu analizi çalışmalarında kullanılmaktadır (Eliaçık ve Erdoğan, 2015).

Duygu Analizi: öznel bilgileri çıkarmak için doğal dil işleme, metin analitiği ve bilişimsel dilbilim alanlarından ödünç alan disiplinler arası bir alandır (Duwairi et al., 2014).

Şirketler bugünlerde ürün ve hizmetlerini tanıtmak için sosyal medyayı kullanıyorlar. Birçok şirket müşterileri ile iletişimde kalmak için Facebook ve Twitter hesaplarını kullanmaktadır. Müşteriler de aynı zamanda ürünler ve hizmetler hakkında bilgi almak için de sosyal medyayı kullanıyorlar. Son zamanlarda internet ve özellikle sosyal medya birçok yönden müşterilerin mal ve hizmetler için alışveriş yapma şeklini değiştirdi. Müşteriler artık ürün almadan önce çevrimiçi (online) tüketici yorumlarını okuyarak başka müşterilerin farklı ürünlerin güçlü ve zayıf yönleri hakkında yaptıkları yorumları dikkate alıyorlar. Bu şekilde farklı tüketicilerin fikirleri insanların ihtiyaçlarına en uygun olanları bulmalarına yardımcı olmaktadır. Satıcı tarafından oluşturulan ürün açıklamaları ile karşılaştırıldığında, çevrimiçi (online) tüketici yorumları daha fazla kullanıcıya yöneliktir ve ürünü farklı kullanım senaryoları açısından tanımlar ve farklı kullanıcıların bakış açısından değerlendirir. İnternet üzerinden büyük miktarlarda verilere erişme ve bunun bir sosyal ağa dönüşmesi artık bir sorun değil, çünkü web'de her gün insanların kullanımına sunulan yeni bilgiler terabaytlarca üretiliyor. Daha da önemlisi, bilgiyi paylaşma biçimimiz değişti. Sosyal medya kullanımı gün geçtikçe artmakta olduğu için bu alanda olan kullanıcıların internet üzerinden artması aynı anda çeşitli tartışma ve etkinliklere katılımlarını da arttırmıştır. Bir ürün bazında kullanıcıların yorumları ürün hizmetleri hakkında birçok önemli kararın alınmasına yardımcı olmaktadır. Ancak bu kadar büyük miktarda yorumu miktar yorumlarını manuel olarak okumak ve karar almak çok zor bir iştir. Dolayısıyla, ürünün olumlu ve olumsuz özelliklerini otomatik olarak çıkarmak ve karar alma sürecini kolaylaştırmak için otomatik bir sisteme ihtiyaç vardır. Günümüzde bu faaliyetleri gerçekleştiren birçok site ve şirket vardır.

Duyarlılık analizi aynı zamanda metin içinde mevcut bilgilerin çıkarılmasıyla ilgilenen bir metin sınıflandırma problemidir. Duyarlılık analizi ayrıca görüşün öznelliğini veya nesnelliğini bulmakla da ilgilidir. Öznellik ve nesnellik nedir? Öznellik, birinin kişisel incelemesi ile ilgili iken, nesnellik bir uzman tarafından verilen görüştür (Vidushi and Sodhi, 2017).

Duygu Analizi (DA) veya Fikir Madenciliği (FM), insanların bir varlığa yönelik görüş, tutum ve duygularının bilişimsel bir çalışmasıdır. Varlık bireyleri, olayları veya konuları temsil edebilir. İki ifade DA veya FM birbiriyle değiştirilebilir. Bunlar birbirine yakın kavramlar olsa da son zamanlarda araştırmacılar bunların farklı olduğunu belirtmektedirler. FM insanların bir varlık hakkındaki düşüncelerini analiz edip onu açığa çıkarır. Diğer yandan DA ise bir metni analiz ederek oradaki duygusal ifadeleri ortaya koyar (Can və Alatas, 2017).

2.1.Duygu Analizi Gerekliliği

Fikirler hemen hemen insanların faaliyetlerinin merkezinde yer almaktadır, çünkü davranışlarımızın kilit unsurudur. Ne zaman bir karar vermemiz gerekiyorsa, başkalarının da bu konu hakkında görüşlerini bilmek istiyoruz. Gerçek yaşamda işletmeler veya kuruluşlar her zaman ürünleri ve hizmetleriyle ilgili tüketici veya halkın düşüncelerini bilmek isterler. Ayrıca bireysel tüketiciler de bir ürünü satın almadan önce başka tüketicilerin görüşlerini veya bir siyasi seçimde oylama kararı vermeden önce diğer insanların siyasi adaylarla ilgili görüşlerini bilmek isterler. Geçmişte, bir kişi fikirlere ihtiyaç duyduğunda arkadaşlarına ve ailesine soruyordu. Bir kuruluş veya işletme ise kamu veya tüketici görüşlerine ihtiyaç duyduğunda anketler, kamuoyu yoklamaları ve odak grupları oluşturuyorlardı. Bu görüşleri edinmek ise pazarlama, halkla ilişkiler ve politik kampanya şirketleri için büyük bir iş yükü olmuştur. Web ‘de sosyal medyanın (örneğin: incelemeler, forum tartışmaları, bloklar, mikro bloklar, Twitter, yorumlar ve sosyal ağ sitelerindeki yorumlar) patlayan büyümesiyle, bireyler ve kuruluşlar bu medyadaki içeriği giderek daha fazla karar almak için kullanmaya başladılar. Günümüzde, eğer tüketici bir ürünü satın almak istiyorsa, artık bunu almadan önce araştırmasına arkadaşları ve ailesinin görüşleri ile sınırlı kalmıyor, çünkü web ’de halka açık forumlarda ürünle ilgili çok sayıda kullanıcı yorumları ve tartışmalar vardır. Artan veri miktarından dolayı firmaların kamuoyu görüşlerini toplamak amacıyla anketler, kamuoyu yoklamaları ve odak gruplarının oluşturmasına gerek göremiyorlar, çünkü mikro blok sitelerinde kamuya açık bilgi bolluğu vardır. Bununla birlikte, Web ’deki fikir sitelerinin bulunması, izlenmesi ve içerdiği bilginin çıkarılması farklı sitelerin olmasından dolayı zorlu bir görev olmaya devam etmektedir. Her site genellikle uzun bloklar ve forum içeriği

şeklinde kolayca deşifre edilmeyen çok büyük miktarda fikir metni içerir. Ortalama bir insan okuyucusu ilgili siteleri tanımlamakta ve içindeki fikirleri almakta ve değerlendirmekte zorluk çekecektir. Bundan dolayı otomatik duyarlılık analiz sistemlerine ihtiyaç vardır (Liu, 2015).

İnternet üzerinden büyük miktarda yorum ve tartışma okumak ve bunun sonunda karar almak kolay bir iş değildir (Vidushi and Sodhi, 2017). Otomatik duyarlılık analiz araçlarının mevcudiyetinden önce, müşterilerin değerlendirmelerini alma süreci oldukça zahmetli ve zaman alıcı bir işti (Duwairi et al., 2014). Geleneksel ölçüm araçlarıyla eşleştirildiğinde, duyarlılık analizi daha zengin bir portre oluşturmamıza ve gelecekteki sosyal medya ve içerik stratejilerinizi bilgilendirmemize yardımcı olabilir.

Tüketiciler kullandıkları ürünler ve servislerin kötü taraflarına sosyal medyada veya farklı mikro blok sitelerinde tepki gösteriyor. Bu nedenle firmalar tüketici davranışının analizini, ürün veya servislerin temel yeterlilik ve yeteneklerin ortaya çıkarmasına yardımcı olan genel planlama ve karar alma işlevlerini yapan otomatik aletlerin kullanılmasının kaçınılmaz ve kritik olduğu görmektedir (Vidushi and Sodhi, 2017).

İngilizce veya başka diller için birçok duyarlılık analiz aracı geliştirilmiştir. Bununla beraber bu tez çalışmasına Azerbaycan dili için bu alanda bulunan eksikliği kapatmak için bir adım atmak hedeflenmiştir.

2.2.Duygu Analizi Kullanım Alanları

Son on yılda ticari ürünler için duyarlılık analizi (fikir madenciliği olarak da bilinir) teknikleri başarıyla kullanılmıştır. Bu yıllar boyunca internet kullanımının katlanarak artışı ile insanlar otel, ürün, siyaset, sağlık ve günlük kullanılan farklı hizmetlere kadar farklı konularda bilgi vermeyi ve paylaşmayı tercih etmektedirler. Bunun yanı sıra insanlar karar vermeden önce başkalarının görüşlerini bilmeyi tercih ettiklerinden sosyal medya fikir madenciliği uygulamaları popülerlik kazanmıştır. Duygu analiz uygulamaları işletmeler tarafından, özellikle de müşterilerin geri bildirimlerinin kritik olduğu hizmet

endüstrisinde yoğun olarak kullanılmaktadır. Aşağıdaki bu uygulama alanlarının bazıları ifade edilmiştir (Duwairi et al, 2014).

Pazarlama: Sosyal medya müşteri etkileşimleri için benzersiz bir platform haline geldiğinden, duyarlılık analizinin kullanımı pazarlamayı tamamen yeni bir seviyeye taşımıştır. Pazarda kalabilmek için müşterilerin senin ve rakiplerinin ürünleri veya hizmetleri hakkında neler hissettiğini belirlemekle ilgilidir. Şirketler, tüketicilerin sosyal medya paylaşımlarında kullanıldıkları duyguları onların marka imajını şekillendirdiğini fark etti. Bundan dolayı şirketlerin yeni ürünleri hakkında doğru ve zamanında bilgi alması bu üreticilere rekabet avantajı ve yeni ürün geliştirme konusunda yardımcı olur. Bu nedenle, duyarlılık analiz araçları pazarlamacılara etkinliklerini ölçmek için bir yol sunar veya bir ürün veya hizmeti araştırmaya çalışan tüketicilere yardımcı olur. Firmalar duyarlılık analizi ile tüm olumsuz geri bildirimleri tespit etme ve ona zamanında tepki verme fırsatına sahip olurlar ve böylece onlar kendi marka değerlerini koruyabilirler. Benzer şekilde olumlu geri bildirimleri tespit edebilir ve bu olumlu geri bildirimi yapan tüketicilere teşekkür edilebilir ve böylece işletme ile müşteriler arasındaki bağı daha da güçlendirilebilir. Bu tüketicilere yeni bir ürün tanıtmak ve sunmak için harika bir yol olabilir.

Siyaset: Siyasi kuruluşlar insanların sosyal medya paylaşımlarındaki görüşlerin üzerinde duyarlık analiz ederek birçok değerli bilgi elde edilebilir. Bu analizleri kullanarak siyasiler politikaları ile ilgili potansiyel tehdit, sorun ya da sorumlular hakkında bilgi sahibi olabilmektedir. Buna ek olarak siyaset kuruluşları aldıkları kararların insanların hayatını nasıl etkileyeceği hakkında bilgi sahibi olmak ister. Örneğin fiyatların artırılması veya yasanın değiştirilmesi gibi önemli çalışmaların topluma nasıl yansıması olacağı hakkında geri bildirimin alınması konusunda duyarlılık analizi önemli bir rol oynamaktadır. Bu bilgileri kullanarak yapılacak seçim sonuçlarının öngörülmesinde siyasi kuruluşlar veya siyasetçiler görüş sahibi olabilmektedir.

Sağlık hizmeti: Teknolojinin gelişmesi ile birlikte insanlar sağlık sorunlarını çevrimiçi ortamda paylaşıyor ve daha önceki tecrübelerinden ve ailesinden öğrendikleri yöntemler hakkında tavsiyede bulunuyorlar. Bu sorunlar

hakkında verileri bloklar, forumlar veya çok çeşitli konuları kapsayan sosyal medya siteleri gibi çeşitli kaynaklarda yayınlanmaktadır. Bu kanyaklarla birlikte insanların sağlık sorunlarını, hastalıklarını, ilaçlarını tartıştıkları sağlıkla ilgili özel bloklar ve forumlar vardır. İnsanlar için hastalıkları, ilaçları ya da bir sağlık merkezi seçimi ile ilgili karar alma konusunda başkalarının edindikleri deneyimleri öğrenmek çok değerlidir.

Çevrimiçi olarak sunulan bu içerikler ücretsiz ve büyük miktardadır, bu nedenle tüm bu bilgileri manuel olarak analiz etmek ve bunları hızlı ve etkili bir karara doğru sonuçlandırmak çok kolay değildir. Duyarlılık analizi teknikleri bu görevi en az kullanıcı desteğine sahip olan veya hiç olmayan otomatik işlemlerle gerçekleştirmektedir. Bunun sonucunda insanlar ilgilendiği sağlık sorunu ile ilgili kararlarının önerilen veya önerilmeyen olmak üzere iki sınıfa ayrılması şeklinde olmaktadır.

Finans: Duygu Analizi, finansal dünyada da kullanılabilir. Yatırımcılar ilgilendiği şirketleri kolayca takip edebilir ve duygu verilerini gerçek zamanlı olarak izleyebilirler. Duyarlılık analizi ile işletme yatırımcıları iş haberlerini daha kolay alabilir ve daha iyi finansal kararlar almak için bu bilgileri kullanabilirler.

Gerçek hayatta, finansal piyasa analistleri borsadaki haberlere ve olaylara dayanarak tahminlerde bulunurlar. Benzer şekilde, Duyarlılık Analizi uygulamaları insanların yerine aynı işi yapmaktadır. Dahası, fikir madenciliği uygulamaları dilbilim ve ileri makine öğrenme teknikleri kullanarak birkaç dakika içinde çeşitli haber kanallarında büyük miktarda metin tarama yeteneğine sahip insan analistlerinden daha verimli olduğunu kanıtladı.

2.3.Duygu Analizinin Zorlukları

Dil evrendeki en harika, dinamik ve gizemli olgudur. Dil ve onun yapısı birincil zorluktur. Duygu analiz alanında birkaç açık uçlu problem ve zorluklar vardır. Bunlardan bazıları Kolkur ve arkadaşları aşağıdaki gibi belirtmiştir (Kolkur et. al., 2015).

Kelime anlam bozukluğu: Sıklıkla karşılaşılan sorun kelime anlam bozukluğudur. Kelimelerin farklı alanlar için farklı anlamları olabileceğinden, içeriğe dayalı kelimenin doğru anlamı çıkarılmalıdır. Örneğin, "küçük" cep telefonları için olumlu bir görüş ifade edebilirken, ancak bu kelimeyi otel odası için kullandığımızda olumsuz olabilmektedir (Kolkur et. al., 2015.

Dilbilgisi hataları ve kötü yazımlar: Duyguları analiz eden birçok yaklaşım vardır ancak dilbilgisi hataları ile ilgili yapılan çalışmalar neredeyse hiç yoktur. Kullanıcılar her zaman bir yorum yazarken tam gramer kurallarına, noktalama işaretlerine veya başka yazım kurallarına uymayabilir. Bu hatalar bağlamı farklı şekillerde anlamaya neden olur. Örneğin "Bu araba çokkkk güzelllllllll", cümlesinde kullanıcı aslında "çok" ve "güzel" yerine daha derin bir duygu göstermek için "çokkkk, ve "güzelllllllll" kelimelerini kullanmıştır. Bu tanım insanlar tarafından doğru bir şekilde yorumlanabilse de ancak sistem "çokkkk, ve "güzelllllllll" olarak yazılan kelimeleri farklı yorumlayacaktır.

Artımlı yaklaşım: Gerçek zamanlı verilerin analizi tek seferlik bir işlem değildir. Ne zaman veri eklenirse analiz yapmamız gerekiyor, o zaman neden önceki analiz sonucunu kullanmamalıyız sorusu çıkıyor. Artımlı yaklaşım, var olan bir sonucun geçmiş örnekleri yeniden işlemeden sadece yeni gelen veri örnekleri kullanarak analiz sonucunu güncellenmesine izin verir. Bu veriler zaman içinde değiştiğinde tüm veri kümesinin bulunmadığı durumlarda yararlı olabilir (Patil and Atique, 2015).

Gürültü ve Dinamizm yönetimi: Sosyal medya verileri muazzam derecede büyük, gürültülü, yapılandırılmamış ve dinamiktir, bu nedenle yeni zorluklar ortaya çıkmaktadır. Gürültülü verilerinin belirlenmesi ve silinmesi zorlu bir iştir (Patil and Atique, 2015).

Karşılaştırmalar: Karşılaştırmalı cümleler için polariteyi belirlemek kolay değildir. Örneğin, X telefonunun pil ömrü, Y telefonlundan daha iyidir. Bu metinde "daha iyi" olarak kullanılan bir olumlu kelime var, ancak yazarın tercih ettiği nesnenin karşılaştırmalı olduğundan anahtar varlığın(nesnenin) hangisi olduğunu belirlemek kolay değildir (Kolkur et. al., 2015).

Olumsuzluk: Doğru şekilde ele alınmadığı takdirde yapılan olumsuzluklar tamamen yanlış sonuçlar verebilir. Örneğin, “bu telefonun kolayca kırılmaması için iyi bir şans var”. Bu örnek olumlu polarite gösteriyor ancak olumsuzlaşmanın olması etkiyi tamamen değiştiriyor.

İstihza (Sarkasım): Bir başka ilginç zorluk, alaycılığın tanımlanması ve metinde ifade edilen duyguların daha ince taneli bir seviyede analiz edilmesidir. İstihza bir konu veya olay hakkında karşı tarafı incitmek veya rahatsız etmek için kullanılır. Bazı durumlarda ise komik etki yaratmak için kullanılabilir. “Çocuklar gerçekten bir evi aydınlatıyor - ışıkları hiç kapatmıyorlar” bu örnekte çocuklar için kelime olarak olumlu kelimeler kullanılsa da anlam olarak ise bu cümle olumsuz bir duygu belirtmektedir. İstihzanı ifadelerden belirlemek ve doğru içerik ilgili duyguları bulmak zor bir iştir. Birini veya bir şeyi övüyor gibi görünen ama gerçekte onunla alay etmek veya iğnelemek için kullanılan ironi veya satirik sözlerden oluşur (Patil and Atique, 2015).

Cambria'ya ve arkadaşlarına (2013) göre doğal dilden fikir ve duygu madenciliği yapmak oldukça zordur çünkü açık ve örtülü, düzenli ve düzensiz, sözdizimsel ve anlamsal dil kurallarının derinlemesine bilinmesi gerekiyor. Duygu analizi araştırmacıları aynı zamanda yukarıda bahsi ettiğimiz NLP’nin çözülmemiş olumsuzlaşma yönetimi (negation handling), adlandırılmış varlık tanıma ve kelime-anlam belirsizliği gibi problemleriyle mücadele ediyor. Fikir madenciliği çok kısıtlı bir NLP problemidir, çünkü sistemin yalnızca her cümlenin veya konuların olumlu veya olumsuz duygularını anlaması gerekiyor. Bu nedenle duygu analizi NLP araştırmacılarının NLP'nin tüm cephelerinde somut bir ilerleme kaydetmeleri ve potansiyel olarak büyük bir pratik etkiye sahip olmaları için bir fırsattır.

Liu DA`nın temsil ettiği çok yönlü problemin temel teknik zorluklarını ise aşağıdaki başlıklar arasında bulunabileceğini iddia ediyor

(1) Yesterday, I bought a Nokia phone and my girlfriend bought a moto.
(Dün ben bir Nokia telefonu ve kız arkadaşım bir moto aldı)

(2) We called each other when we got home. (Eve gidince birbirimizi aradık)

(3) The voice on my phone was not clear. (Benim telefonumdaki ses net değildi)

(4) The camera was good. (Kamera iyiydi)

(5) My girlfriend said that the sound on her phone was clear. (Kız arkadaşım telefonundaki sesin açık olduğunu söyledi)

(6) I wanted a phone with good voice quality.(Ses kalitesi iyi olan bir telefon istedim.)

(7) So I was satisfied and returned the phone to BestBuy yesterday (Bu yüzden memnun oldum ve dün telefonu BestBuy'a iade ettim.)

Nesne tanımlama: Nesnenin ne olduğunu ve hangi fikrin verildiğini keşfetmektir. Örnek olarak kullanılan paragrafta, nesneler Motorola`nın kısaltılmışı olan "moto" ve Nokia'dır. "BestBuy" ismi ise mağazanın adına karşılık gelir; bu nedenle inceleyen kişinin sağladığı işlenen karşılaştırmanın bir parçası değildir ve ürünlerin karşılaştırması açısından bir nesne değildir (Appel et al., 2015).

Özellik çıkarma ve eş anlamlı gruplandırma: Örnekte bahsedilen özellikler "ses" ve "kamera"dır. Liu'ya göre “Bu sorunu çözme girişimleri olsa da, halen büyük bir sorun olmaya devam ediyor”. Buna ilave olarak, bir özellik farklı şekillerde ifade edilebilir. Örneğin, “voice” (ses) ve “sound”(ses) yukarıdaki örneğimizde aynı özelliği ifade eder (Appel et al., 2015).

Görüş oryantasyonu sınıflaması: Bu görevin amacı, verilen bir cümlenin bir özelliği hakkında görüş olup olmadığını bulmaktır. Varsa olumlu mu, olumsuz mu, tarafsız mı?. Kilit konulardan biri, duyarlılık analizinde etkili olan fikir sözcüklerini ve cümleleri tanımlamaktır. Sorun, insanların fikirlerini ifade etmek için kullandıkları sınırsız sayıda ifade olduğu ve farklı alanlarda bu önemli ölçüde

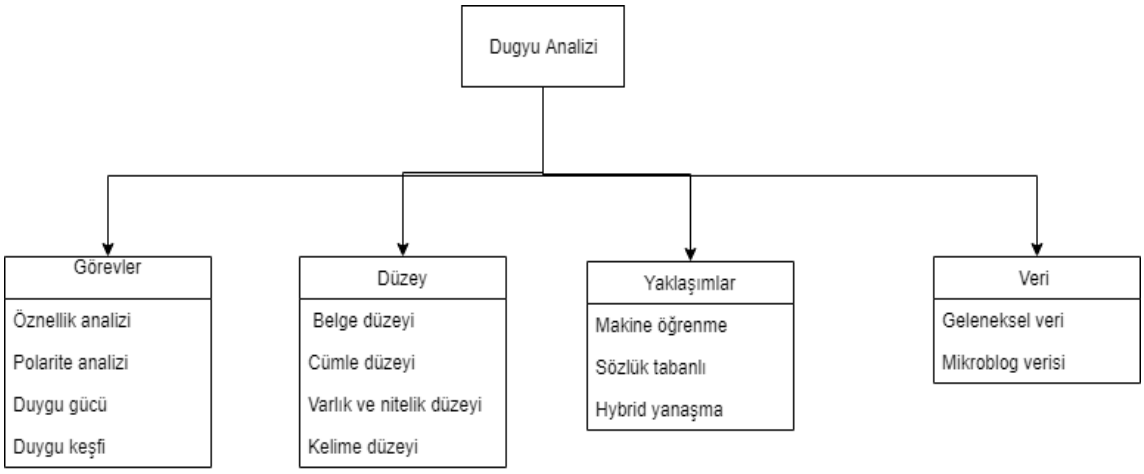
farklı olabilmesidir. Aynı alanda bile, aynı kelime farklı şartlarda farklı görüşler gösterebilir (Appel et al., 2015).

Bütünleşme: DA'nın temel amacı bir belgeye girdi olarak verilen tüm beşli $(e_j, a_{jk}, so_{ijkl}, h_i, t_1)$ parçaları keşfetmektir ama bu karmaşıktır, çünkü beşlikteki beş bilgi birbiriyle eşleşmelidir.

2.4.Duygu Analizinin Ana Unsurları

Fikir madenciliği ve duygu analizi çok zengin bir araştırma problemidir ve her yıl bu problemin çeşitli yönlerini ve boyutlarını hedef alan çok sayıda çalışma yayınlanmaktadır. Genel olarak belirtirsek, duyarlılık analizi sorunu, Şekil 1'te gösterildiği ve aşağıda özetlendiği gibi dört ana boyuta bölünebilir.

1) **Görevler:** Bir sonraki alt bölümlerde açıklanacağı gibi kutup(polarite) algılama, duygu algılama ve duygu gücü gibi daha ince taneli görevlere kadar değişen duyarlılık görevleri.
2) **Analiz düzeyi:** Sözcük, ifade, cümle ve belge olmak üzere farklı düzeyde analiz yapabilmektedir.
3) **Yaklaşımının türü:** Denetimli, sözlük temelli, hibrid ve başka yaklaşım türleri kullanılmaktadır.
4) **Veri tipi:** Duygu geleneksel metinlerden (örneğin, haberler, makaleler, ürün incelemeleri) veya mikro blok verilerinden (tweet mesajları, Facebook durum güncellemeleri, SMS, vb.) çıkarılabilir.



Şekil 2.2. Duygu analizin boyutu

Duygu analizinin farklı boyutlarını göstermek için iPhone telefonu ile ilgili aşağıdaki metini örnek olarak kullanacağız. Örnekteki her cümle kolay referans verebilmek için bir sayı ile birleştirilmiştir (Saif 2015).

(1) I bought an iPhone a few days ago (Birkaç gün önce bir iPhone aldım).

(2) It was such a nice phone (O çok hoş bir telefon oldu).

(3) The touch screen was really cool (Dokunmatik ekran gerçekten harikaydı).

(4) The voice quality was clear too (Ses kalitesi de açıktı).

(5) Although the battery life was not long, that is ok for me (Pil ömrü uzun olmamasına rağmen, bu benim için sorun değil).

(6) However, my mother was mad with me as I did not tell her before I bought it (Ancak, satın almadan önce anneme söylemediğim için o bana kızmıştı).

(7) She also thought the phone was too expensive, and wanted me to return it to the shop (Ayrıca telefonun çok pahalı olduğunu düşündü ve mağazaya geri vermemi istedi.).

Bu metinde fark edebileceğimiz ilk şey çeşitli fikirlerin olduğudur. Yukarıda verilen örnek metinde (2), (3) ve (4) numaralı cümlelerde olumlu fikirler ifade ederken (5), (6) ve (7) numaralı cümleler ise olumsuz fikir ve duyguları ifade edilir. Daha sonra fikirlerin hepsinin ifade edildiği bazı hedeflerin veya nesnelerin olduğunu fark ettik. (2) Numaralı cümlede fikir bir bütün olarak iPhone olurken (3), (4) ve (5) numaralı cümlelerde ise fikirler ise sırası ile "dokunmatik ekran", "ses kalitesi" ve "pil ömrü" olarak belirtilmiştir. Cümle (7) içindeki görüş iPhone'un fiyatı ile ilgiliyken ancak cümle (6) içindeki fikir veya duygu iPhone değil "ben" ile ilgilidir. Bu önemli bir noktadır. Bir uygulamada kullanıcı belirli hedefler veya nesneler hakkındaki görüşlerle ilgilenebilir ancak hepsine bakmayabilir (örneğin yukarıdaki metinde "ben" ile ilgilenmeyebilir ama "iPhone" ile ilgilenir). Son olarak, fikirlerin kaynaklarını veya sahiplerini de görebiliriz. (2), (3), (4) ve (5) numaralı cümlelerdeki görüşlerin kaynağı veya sahibi değerlendirmenin yazarıdır ancak (6) ve (7) cümlelerinde ise fikir sahibi yazarın annesidir. Bu örnek göz önünde bulundurularak artık duygu analizi veya fikir madenciliği problemi tanımlanabilir (Liu, 2010).

Duyarlılık analizinin temel alt görevi polarite tespitidir. Belirli bir metnin duyarlılığının olumlu veya olumsuz olmasını öğrenmek isteriz. Örneğimizde, cümle (2),(3) ve (4) olumlu, cümle (5) ise olumsuz bir duyguya sahiptir. Bir başka popüler alt görev metnin öznel yani olumlu veya olumsuz bir duyguya sahip olduğunu veya nesnellik(objektif) yani tarafsız bir duyguya sahip olup olmadığını tanımlamayı amaçlayan öznellik tespitidir. Yine yukarıdaki örneğimizi temel alırsak cümle (1) nesnel cümlelerdir veya nötr. Öte yandan (2), (3), (4), (5), (6) ve (7) cümlelerinin tümü özneldir.

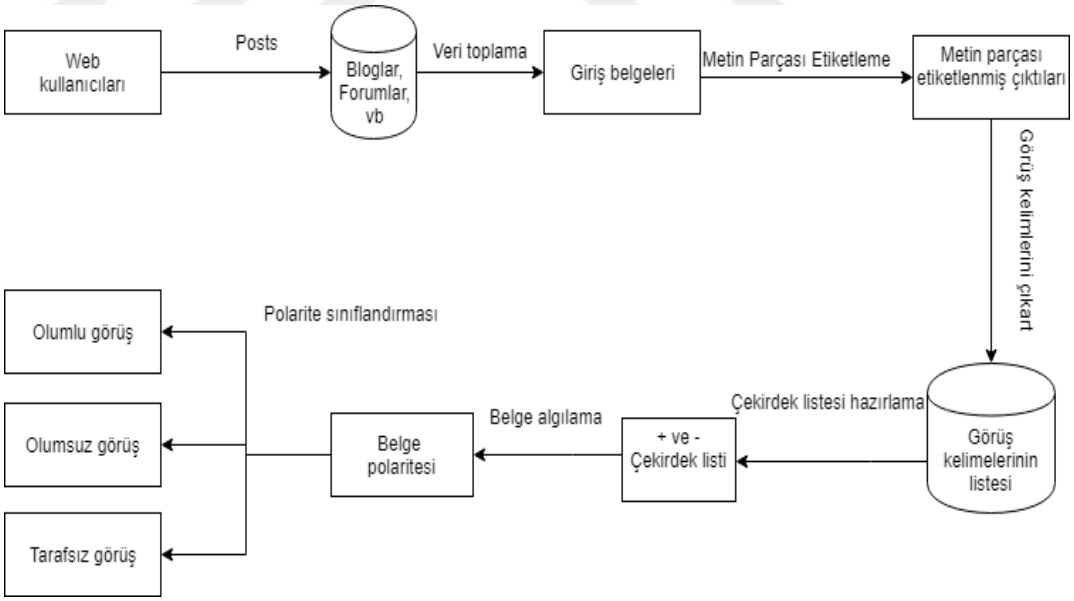
Öznellik ve kutupluluk(polarite) tespiti, büyük ölçüde duyarlılık analizinde en popüler alt görevlerdir ancak yalnızca bunlar değildir. İlgili diğer alt görevlerden biri "mutluluk", "üzüntü", "öfke", vb. gibi metinle ifade edilen insan duygularını amaçlayan duygu tespitidir. Örneğimizde cümle (2) "hoş" ve cümle (6) da ise "harika" duyguları gösterilmektedir. Metindeki duygunun kuvvetini veya gücünü ölçmeyi amaçlayan bir başka alt görevlerden duygu gücü tespitidir. Bu görev cümle içindeki olumlu duygunun ne kadar güçlü olduğunu veya

olumsuz duygunun cümle içinde ne kadar güçlü olduğunu tespit etmeye çalışır. (Liu, 2010).

2.5.Duygu Analizinin Düzeyleri

Duyarlılık analizi temel olarak dört düzeyde incelenmektedir: Kelime, ifade, cümle ve belge düzeyi. Duygu düzeyindeki seçim, duyguların metinde ifade edilme şeklinden kaynaklanmaktadır.

Belge düzeyi: Bu sınıflandırmanın en basit şeklidir. Belge düzeyinde duyarlılık sınıflandırmadaki ana görevi belgenin genel duyarlılık yönelimini belirlemektir. Bu seviye görüş belgesini olumlu, olumsuz veya tarafsız düşünceye göre sınıflandırır (Jagtap and Pawar, 2013). Ama bu yaklaşımda belgenin sadece tek bir nesne (film, kitap veya otel) hakkında görüşü içerdiği varsayılmaktadır. Forumlarda ve bloklarda olduğu gibi belge farklı nesneler hakkında görüş içeriyorsa bu yaklaşım uygun değildir. İşlemden önce ilgisiz cümleler kaldırılmalıdır (Kolkur et. al., 2015).



Şekil 2.3. Belge düzeyi

Cümle düzeyi: Adından da anlaşılacağı gibi, tüm belgede yapılmak yerine cümleler üzerinde duyarlılık analizi yapıldığına işaret etmektedir**.** Cümle düzeyinde sınıflandırmada her cümleyi ayrı bir birim olarak kabul eder ve

cümlenin yalnızca bir görüş içermesi gerektiği düşünülür. Eğer cümle karışık bir kutupluk içeriyorsa (çok özellikten dolayı hem pozitif hem de negatif) bileşenlerin nispi duyarlılık kuvvetlerine dayanarak genel bir kutupluluk atanır. Bir belgeyi doküman seviyesinden daha ayrıntılı bir seviyede incelemek için cümle seviyesi kullanılabilir. Bu seviye her cümlenin olumlu, olumsuz veya tarafsız cümle düzeyinde bir duyarlılık gösterdiğini belirler. Bu düzeydeki duyarlılık analizinin öznellik sınıflaması ve duyarlılık sınıflaması olarak iki görevi vardır.

Öznellik sınıflama: Bir cümle öznel (sübjektif) veya nesnel (objektif) olabilir. Nesnel cümle gerçekleri içerir. Buna karşılık öznel cümleler ise nesne veya varlık hakkında eleştiri veya görüşler içerir. *"Azerbaycan'ın ekonomisi büyük ölçüde petrol ve doğalgaz endüstrisine bağımlıdır. Bakü yaşamak için mükemmel bir yerdir*". Bu metin parçasında kullanılan ilk cümle gerçek veya nesnel bir cümledir ve Azerbaycan'a karşı herhangi bir duygu ifade etmemektedir. Bu nedenle, metnin kutupsallığına karar vermekte hiçbir rol oynamamaktadır ve filtrelenmelidir. İkinci cümle ise "Bakü" hakkında olumu bir görüş belirtmektedir, dolayısı ile öznel bir cümledir. Cümle düzeyinde analizinin avantajı öznellik-nesnellik sınıflamasını yapabilmesidir.

Duyarlılık sınıflama: Cümle içinde bulunan görüş kelimelerine bağlı olarak olumlu, olumsuz veya tarafsız olarak sınıflandırılabilir. Birçok araştırmacılar metni etkili bir şekilde nasıl sınıflandıracağınıza odaklanmaktadır. Belge düzeyinde sınıflandırma yöntemlerinin aynısı cümle düzeyinde sınıflandırma problemine uygulanabilir.

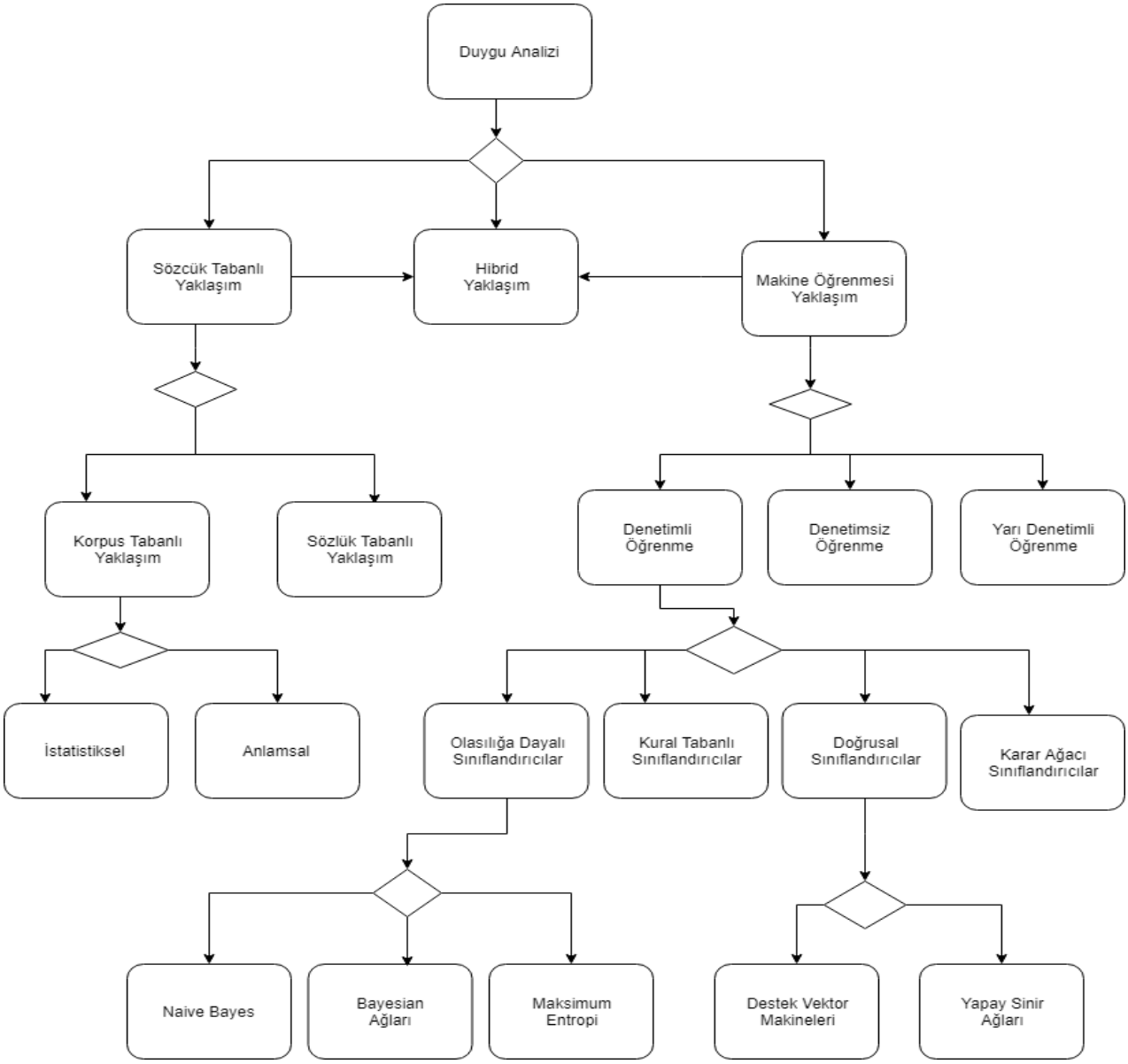
Varlık ve Nitelik düzeyi: Hem belge düzeyi hem de cümle düzeyi analizi, insanların tam olarak neyi sevip neyi sevmediğini keşfetmez. Varlık düzeyinde ise ince taneli analiz yapılır. Bu analizin temeli bir düşünce; duyarlılıktan (olumlu veya olumsuz) ve hedeften oluşur fikrine dayanıyor (Kiruthika.et. al, 2016). Varlık düzeyinde duyarlılık analizi aynı zamanda özellik tabanlı duyarlılık analizi olarak da adlandırılır (Shirsat. et. al., 2017). Özellik düzeyi duyarlılık analizinin sonuçları, kararlar almak için hem müşteriler hem de üreticiler için çok faydalıdır. Müşteriler satın alma kararını vermeden önce farklı özelliklere dayalı ürünleri karşılaştırmak amacıyla bu düzeyde analizin sonuçlarını kullanabilir. Benzer

şekilde, üreticiler ürünlerini geliştirmek ve farklı pazarlama stratejileri başlatmak için ürünün güçlü ve zayıf yanlarını bilmekten dolayı özellik seviyesi duyarlılık analizinin sonuçlarını kullanabilirler (Farooq. et. al., 2017). Örneğin "*Telefon öncelikle tasarım harikası. Kamera olarak diyebileceklerim renkler çok canlı, gece fotoğraf çekimleri hem ön hem arka kamera bence çok iyi ama pil ömrü çok kötü ki bence bu fiyata yeter de artar*" cümlesinde bahsedilen telefonun 3 tane özeliği hakkında fikir bildirilmiştir. Bunlardan "tasarım" ve "kamera" hakkında olumlu görüş belirtilirken "pil ömrü" hakkında ise olumsuz görüş belirtilmiştir.

Kelime düzeyi: Kelimeye dair duyarlılık polaritesi atamak çok kolay değildir, çünkü bir kelime farklı alanlarda ve hatta aynı alanda farklı kutuplara sahip olabilir. Örneğin "uzun" kelimesi "*Bu haftaki yazımda; batarya ömrü* ***uzun*** *olan akıllı telefonlardan bahsedeceğim*" cümlesinde telefonun batarya ömrü için olumlu bir polariteye sahipken aynı alanda kullanılan kelime "*bu akıllı telefonun açılma süresi* ***uzun*** *sürüyo*r" cümlesinde ise olumsuz bir polariteye sahiptir. Bu düzeyde genellikle odak sıfatlar aranmaktır. Ancak fiiller, zarflar ve isimler de bir öznellik duygusu taşıyabilir ve görüş bildirebilirler (Appel et al., 2015).

3. DUYARLIK ANALİZ YÖNTEMLERİ

Duygu sınıflandırma teknikleri makine öğrenme, sözlüğe dayalı öğrenme ve hibrid öğrenme olmak üzere 3 ana yaklaşıma ayrılabilir. Makine öğrenme yaklaşımı ünlü MÖ algoritmalarını uygular ve dil özelliklerini kullanır. Sözlük tabanlı yaklaşım ise bilinen ve önceden derlenmiş bir duyarlılık terimleri koleksiyonu olan duyarlılık sözlüğüne dayanır. Hibrid yaklaşım her iki yaklaşımı da birleştirir ve yöntemlerin çoğunda kilit rol oynayan duyarlılık sözlüğü ile bu yaklaşımda çok önemlidir. Bu tekniklerin daha ayrıntılı açıklaması aşağıdaki alt bölümde verilmiştir (Medhat et al, 2014).



Şekil 3.1. Duygu sınıflandırma yöntemleri

3.1.Makine Öğrenmesi Yaklaşımı

Makine öğrenmesi yaklaşımı tamamen otomatik uygulamalarının olması ve geniş web veri koleksiyonlarını yönetebilmesi nedeniyle, fikir madenciliğinde diğer yaklaşımlarından daha pratiktir. Geleneksel yaklaşımlarda bilgisayara (makineye) veri (girdi) ve program sağlıyoruz ve onu işleyip bize çıktı üretiliyor. Ancak makine öğrenmesi yaklaşımında biz veri (girdi) ve sonuç (çıktı) sağlarız ve bilgisayarın (makine) kendisi girdi-çıktı ilişkisine dayanan bir program çıkarır. Makine öğrenimi, eğitim veri setinin toplanmasıyla başlar. Bir sonraki adım da eğitim verilerini kullanarak bir model hazırlanır.

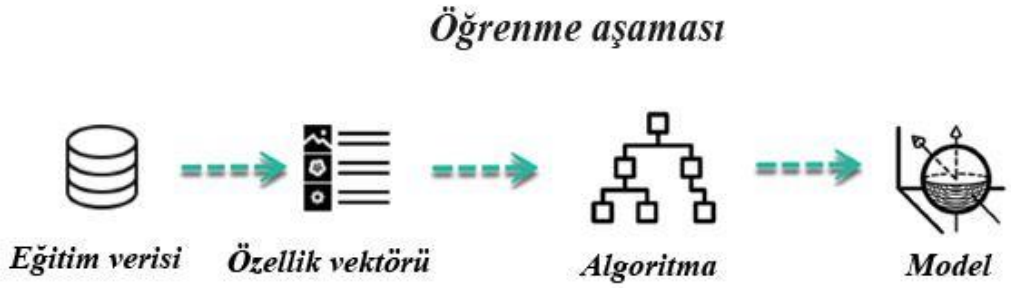
Makine Öğrenimi teknikleri sınıflandırma için bir eğitim seti ve bir test seti kullanır. Eğitim seti, giriş özelliği vektörlerini ve bunlara karşılık gelen sınıf etiketlerini içerir. Bu eğitim seti kullanılarak, giriş özelliği vektörlerini ilgili sınıf etiketlerine sınıflandırmaya çalışan bir sınıflandırma modeli geliştirilir. Ardından, görülmeyen özellik vektörlerinin sınıf etiketlerini tahmin ederek modeli doğrulamak için bir test seti kullanılır.

Makine öğrenmeye dayalı duygu sınıflandırma yöntemleri üç tipe ayrılabilir: denetimli, denetimsiz ve yarı denetimli öğrenme yöntemleri (Madhoushi et al, 2015).

3.1.1. Denetimli öğrenme

Denetimli öğrenme geleneksel lokal sınıflandırmada olgun ve başarılı bir çözümdür ve memnun edici sonuçlar elde edildiği için görüş tespiti için kullanılmaktadır. Denetimli öğrenmeyle ilgili en büyük kısıtlama, eğitim verilerinin miktarına ve kalitesine duyarlı olmasıdır. Eğitim verileri kötü veya yetersiz olduğunda bu yöntem başarısız olabilir. Denetimli sınıflandırma tekniklerinde girdi olarak verilen etiketli deneysel verilere dayanan bir model oluşturur. Bunun sonucunda bir alanı kolayca modelleyebilecek bir sınıflandırıcı oluşturulur. Bu özellikle spam filtreleme gibi konu temelli sınıflandırma görevlerinde yeterli sınıflandırma performansına yol açar. Bununla birlikte

sonuçlar sunulan eğitim verilerinin kalitesi nedeniyle denetimli duyarlılık analiz tekniklerinde değişiklik gösterir(Smith, 2011).



Şekil 3.2. Denetimli öğrenme aşaması

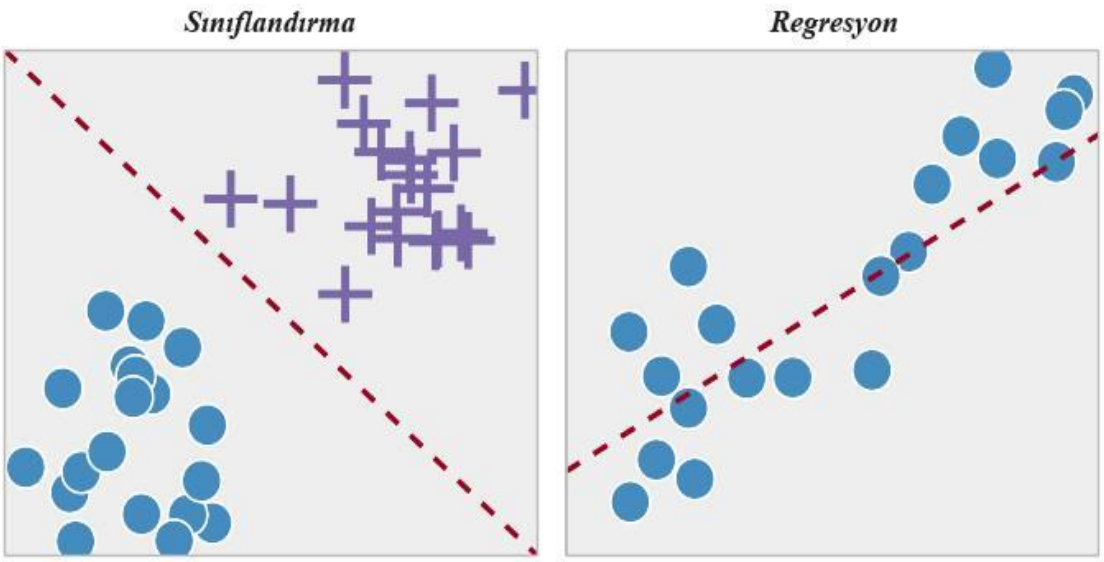
Denetimli öğrenmenin kullanım sebepleri aşağıdaki gibidir:

Denetimli öğrenme, önceki deneyimlerden veri toplamaya veya veri çıktısı üretmeye olanak sağlar.

Önceki deneyimleri kullanarak performans ölçütlerini optimize etmeye yardımcı olur

Denetimli makine öğrenmesi, çeşitli gerçek dünya hesaplama problemlerini çözmeye yardımcı olur.

Denetimli öğrenme temel olarak aşağıdaki gibi iki bölüme ayrılmıştır.



Şekil 3.3. Denetimli öğrenme yöntemleri

Regresyon: Gözetimli öğrenme yöntemlerinden biri de Regresyon yöntemidir. Regresyon problemleri, üretilen çıktının sürekli sayılardan oluştuğu durumlar için kullanılır. Regresyon aynı zamanda bağımlı değişken [Çıktılar] ve bağımsız değişken [Girişler] arasındaki ilişkiyi araştıran bir tahmin modelleme tekniğidir. Bu teknik hava tahmini, zaman serileri modellemesi, süreç optimizasyonu gibi problemlerin çözümünde kullanılmaktadır. Sınıflandırmanın aksine verileri farklı kategorilere sınıflandırmak yerine geçmiş verilere dayanan bir değer tahmin edilmektedir.

Sınıflandırma: Hem insan hem de makine zekâsının özünde yatmaktadır**.** Etiketli verilerin kullanabileceği denetimli öğrenme türüdür ve bu veriler sürekli olmayan bir biçimde tahminlerde bulunmak için kullanılır. Sınıflandırma tekniğinde, algoritma kendisine verilen veri girişinden öğrenir ve ardından bu öğrenmeyi yeni gözlemleri sınıflandırmak için kullanır. Bu veri seti iki sınıflı veya çok sınıflı da olabilir. Sınıflandırma problemi geçmiş verilere dayanarak, öğelerin farklı kategorilere ayrılmasını gerektirir.

Denetimli öğrenme yöntemi olan sınıflandırma alanında birçok öğrenme algoritması vardır. Aşağıdaki makine öğrenim algoritmalarının yaygın olarak

kullanıldığı ve etki alanlarının çoğunda ve farklı veri türlerinde ortalama doğruluk sağladıkları görülmüştür (Desai and Mehta, 2016).

3.1.1.1.Naive Bayes

Naive Bayes sınıflandırıcısı en basit ve en sık kullanılan sınıflandırıcıdır. Bu sınıflandırıcı aynı zamanda metin kategorisinde iyi çalışan basit modeldir. Naive Bayes, olasılıklı bir sınıflandırıcıdır. Bu yöntemde belirli bir özelliğin belirli bir etikete ait olma olasılığını tahmin etmek için Bayes teoremi kullanır.

$$P(sınıf|\text{özellik}) = \frac{P(sınıf)P(\text{özellik}|sınıf)}{P(\text{özellik})}$$

Özellikler, makine öğrenimi sınıflandırıcısına girdisidir. Tweet verileri için ayrıştırılmış tüm kelimeler veya duygu içeren kelimeler özellik olabilir.

P $(sınıf)$, bir sınıfın önsel olasılığı veya rastgele bir özelliğin sınıfı belirleme olasılığıdır.

$P(\text{özellik}|sınıf)$ Verilen bir özellik setinin sınıflandırılması olasılığının önceliğidir.

P (özellikler), belirli bir özellik kümesinin ortaya çıkmasının önsel olasılığıdır (Medhat et al, 2014).

Sınıflandırma çalışmasını basitleştirmek için, Naive Bayes sınıflandırıcı "naif" bir varsayım da yapar, yani tüm özellikler birbirinden bağımsızdır. Böylece, yukarıdaki denklem şu şekilde yeniden yazılabilir:

$$P(sınıf|\text{özellik}) = \frac{P(sınıf) \times P(f1|sınıf) \times P(f2|sınıf) \ldots \times P(fn|sınıf)}{P(\text{özellikler})}$$

f1, f2,… fn'nin her biri bir özelliği gösterir. Naif varsayım asla gerçek olmamasına rağmen, Naive Bayes sınıflandırma sonuçları oldukça iyi olabilir.

Naive Bayes sınıflandırıcısının metin verilerinde nasıl çalıştığını anlamak için iki sınıf kapsayacak pozitif (+) ve negatif (-) duyarlılık analiz çalışmasının örneği verilmiştir. Bunun için 6 satırdan oluşturulan tweet eğitim veri seti ve 1 satırdan oluşturulan tweet test veri seti alınmıştır.

Çizelge 3.1: Eğitim ve test verisine örnek

	Sınıf	Tweet
Eğitim verisi	+	En güzel bayram bu bayram 5-4
	+	Çok güzel bir sempozyum.4-3
	+	Sevgi buna denir 3
	-	Yine bize haram geceler 4
	-	Alıştım yalnızlığın soğuk makamına 4
	-	Hüzünlü bir gün 3-2
Test verisi	?	Bu güzel gecede bizimle olan herkese çok teşekkürler.

Her iki sınıf için önsel olasılıklar P (sınıf) hesaplanır.

$$P_{sınıf} = \frac{N_{sınıf}}{N_{tivits}}$$

Burada $N_{sınıf}$ eğitim verileri içesinde her sınıfın sayısıdır. N_{tivit} ise eğitim verilerinde olan toplam tweetlerin sayısı belirtmektedir (Jurafsky and Martin, 2015).

$$P(+) = \frac{3}{6} = \frac{1}{2} \qquad P(-) = \frac{3}{6} = \frac{1}{2}$$

$$P(w_i|sınıf) = \frac{count(w_i, sınıf) + 1}{\sum_{w \in V}(count(w, sınıf) + 1)} = \frac{count(w_i, sınıf) + 1}{(\sum_{w \in V}(count(w, sınıf)) + |V|}$$

|V|-Eğitim verilerinde olan benzersiz kelimelerin sayısı.

“bu”, “güzel”, “gecede” , “bizimle” , “olan” , “herkese” , “çok” ve “teşekkürler” eğitim verisinde bulunan sekiz kelimenin olasılıkları aşağıdaki gibidir

Çizelge 3.2: Naive Bayes denemesi

$P(\text{bu}\|+) = \frac{1+1}{12+20} = \frac{2}{32}$	$P(\text{bu}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$
$P(\text{güzel}\|+) = \frac{2+1}{12+20} = \frac{3}{32}$	$P(\text{güzel}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$
$P(\text{gecede}\|+) = \frac{0+1}{12+20} = \frac{1}{32}$	$P(\text{gecede}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$
$P(\text{bizimle}\|+) = \frac{0+1}{12+20} = \frac{1}{32}$	$P(\text{bizimle}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$
$P(\text{olan}\|+) = \frac{0+1}{12+20} = \frac{1}{32}$	$P(\text{olan}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$
$P(\text{herkese}\|+) = \frac{0+1}{12+20} = \frac{1}{32}$	$P(\text{herkese}\|-) = \frac{0+1}{11+20} = \frac{1}{3135}$
$P(\text{çok}\|+) = \frac{1+1}{12+20} = \frac{2}{32}$	$P(\text{çok}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$
$P(\text{teşekkürler}\|+) = \frac{0+1}{12+20} = \frac{1}{32}$	$P(\text{teşekkürler}\|-) = \frac{0+1}{11+20} = \frac{1}{31}$

Seçilen tweet T = “bu güzel gecede bizimle olan herkese çok teşekkürler” ifadesi için sınıf olasılık değerleri aşağıdaki şekilde hesaplanır:

$$P(T|+)P(+) = \frac{1}{2} \times \frac{2 \times 3 \times 1 \times 1 \times 1 \times 1 \times 2 \times 1}{32^8} = 5.455 \times 10^{-12}$$

$$P(T|-)P(-) = \frac{1}{2} \times \frac{1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1 \times 1}{31^8} = 2.34 \times 10^{-12}$$

Bu durumda, örnekte verilen tweet `in içeriğinin, olumlu veya olumsuz olma olasılığı belirlendikten sonra test verisinin olumlu olduğu saptanmıştır.

3.1.1.2.Maksimum entropi

Maksimum Entropi, doğal dil işleme alanında yaygın olarak kullanılan bir olasılık dağılım tahmini tekniğidir. Makine öğrenimin diğer öğrenme tekniklerinden farklı değildir, bu yöntemde modelin çıktıları, verilen eğitim veri setine dayanmaktadır. MaxEnt modelleri özellik tabanlı modellerdir. Naive Bayes makine öğreniminin aksine, Maximum Entropy farklı olarak özellikleri için bağımsızlık varsayımlarında bulunmaz. Bununla birlikte, hesaplama açısından daha pahalıdır. Deneysel verilere dayanan bir makine öğrenme yöntemidir.

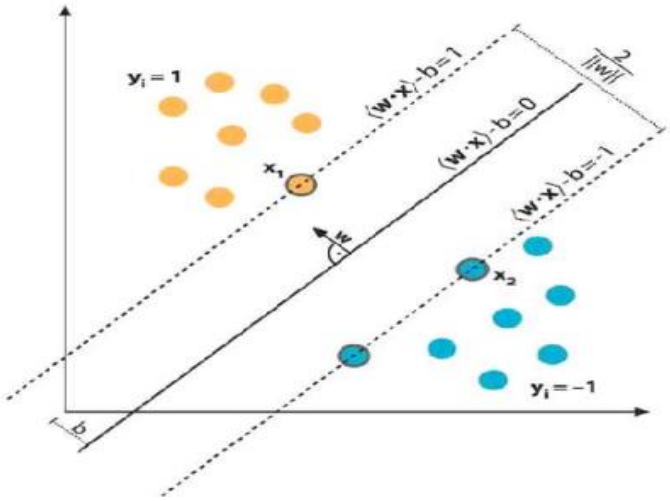
Model aşağıdakilerle temsil edilir:

$$P_{ME}(c|d,\lambda) = \frac{exp[\sum_i \lambda_i f_i(c,d)]}{\sum_x exp[\sum_i \lambda_i f_i(c,d)]})$$

Bu formülde c sınıf, d tweet ve λ ise ağırlık vektörünü temsil etmektedir. Ağırlık vektörleri bir özelliğin sınıflandırmadaki önemine karar verir Daha yüksek ağırlık, özelliğin sınıf için güçlü bir gösterge olduğu anlamına gelir. Ağırlık vektörü, koşullu olasılığı en üst düzeye çıkarmak için lambdaların sayısal optimizasyonu ile bulunur (Go et al, 2009).

3.1.1.3.Destek vektör makinaları

Destek Vektör Makineleri karar sınırlarını tanımlayan karar hiper düzlem kavramına dayanır. Bir karar hiper düzlemi, farklı sınıf üyeliğine sahip veri noktaları arasında ayrım yapan hiper düzlemdir. Şematik bir örnek aşağıdaki çizimde gösterilmektedir. Bu örnekte, veri noktaları ya TURUNCU ya da MAVİ sınıflara aittir. Ayırma çizgisi, sağ tarafındaki tüm veri noktalarının TURUNCU olduğu ve solundaki tüm veri noktalarının MAVİ olduğu bir sınırı tanımlar.



Şekil 3.4. DSM tanımlama

Bu algoritmanın amacı, N-sayısı kadar eğitim verisi verildiği bir uzayda veri noktalarını belirgin bir şekilde sınıflandıran bir karar hiper düzlem bulmaktır. İki veri noktası sınıfını ayırmak için, seçilebilecek birçok olası hiper düzlem vardır. Amaç maksimum ayrıma sahip bir hiper düzlemi, yani her iki sınıfın veri noktaları arasındaki maksimum mesafeyi bulmaktır. Ayrım w vektörü ile temsil edilir. Her x_i örneği p adet niteliğe sahip girdi, $y_i \in \{1, -1\}$ örneklerin ait olduğu sınıfı temsil eden çıktı ve $x \in R^p$ yüksek boyutlu girdi vektörü olmak üzere; x_i, y_i ikililerinden oluşan n hacimli bir eğitim kümesi S verildiğinde, farklı sınıflara ait örnekleri bir birbirinden en iyi şekilde ayıracak,

$$wx + b \ (1)$$

doğrusal hiper düzleminin bulunmasına yardımcı olan denetimli öğrenme algoritmaları sınıfına ait makine öğrenimi algoritmasıdır (Ayan ve Erdoğmuş, 2014).

Eğitim veri setinin doğrusal olarak ayrılabilme durumunda en büyük sınıra sahip ayırma hiper düzlemini bulmaya çalışır. Söz konusu ayırma hiper düzleminin bulunabilmesi için veri setindeki tüm örneklerin

$$f(x_i) = (w_i, x_i) - b \geq 1, y_i = +1 \ (2)$$

$$f(x_i) = (w_i, x_i) - b \leq -1, y_i = -1 \ (3)$$

eşitsizliklerini sağlaması gerekir.

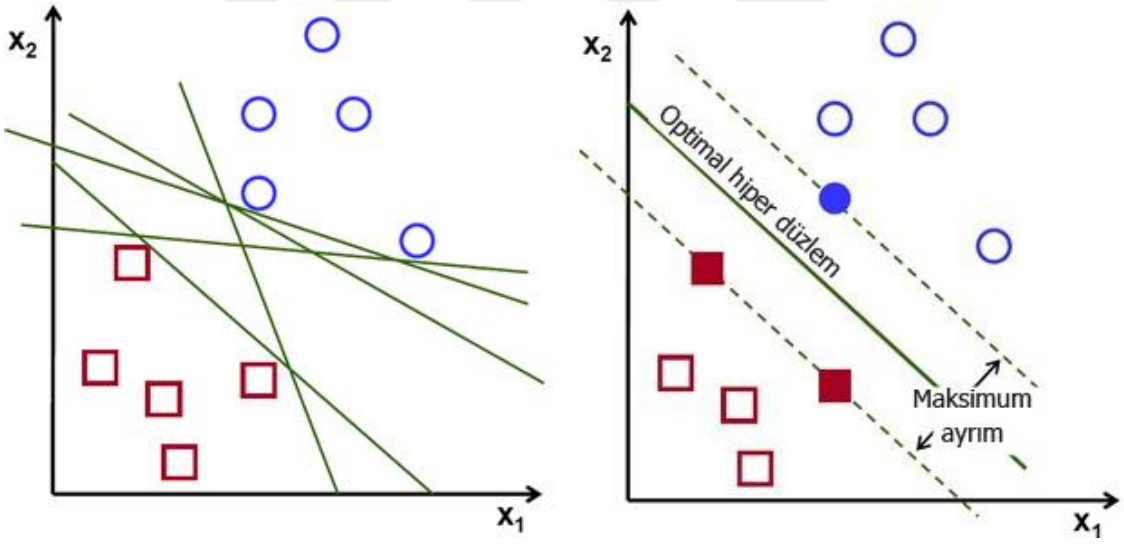
x_i: $w * x - b = 0$ Çoklu düzlemi üzerinde herhangi bir nokta

y_i: Sınıf etiketleri, $y_i \in \{+1, -1\}$

w: Hiper düzlem normali, ağırlık vektörü

b: Sabit

Test örneklerinin sınıflandırılması, veri noktaların hiper düzlemin hangi tarafına düştüklerinin belirlenmesinden ibarettir(Khairnar and Kinikar, 2013).



Şekil 3.5. DSM iki boyutlu hiper düzlem

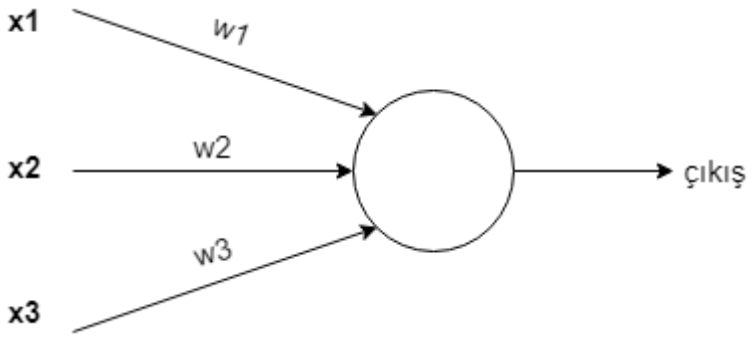
Hiper düzlemler, veri noktalarının sınıflandırılmasına yardımcı olan karar sınırlarıdır. Hiper düzlemin her iki tarafına düşen veri noktaları farklı sınıflara bağlanabilir. Ayrıca, hiper düzlemin boyutu özelliklerin sayısına bağlıdır. Giriş özelliklerinin sayısı 2 ise, hiper düzlem sadece bir çizgidir. Giriş özelliklerinin sayısı 3 ise, hiper düzlem iki boyutlu bir düzlem haline gelir.

Destek vektör makineleri (SVM'ler), geleneksel Naive Bayes 'ten daha iyi performans gösteren geleneksel metin kategorizasyonunda oldukça etkili olduğu gösterilmiştir. Naive Bayes ve Maximum Entropi'nin aksine, olasılıklayıcı değil sınıflandırıcıdırlar. İki kategorili durumda, eğitim prosedürünün arkasındaki temel fikir, maksimum ayrıma sahip bir hiper düzlem bulmaktır.

3.1.1.4.Sinir ağları

Yapay sinir ağları şimdiye kadar icat edilmiş en güzel programlama paradigmalarından biridir. Bu paradigma beyin gibi biyolojik sinir sistemlerinin çalışma biçiminden esinlenen bir bilgi işlem sürecidir. Geleneksel programlama yaklaşımında bilgisayara ne yapacağı söylenilir. Büyük problemler bilgisayarın kolayca gerçekleştirebileceği birçok küçük ve kesin olarak tanımlanmış parçalara ayırılır. Sinir ağlarında ise bunun aksine bilgisayara sorunun nasıl çözüleceği söylenilmemektedir. Bunun yerine gözlemsel verilerden öğrenerek eldeki probleme kendi çözümünü bulmaktadır. YSA'lar da öğrenme biyolojik sistemlerde olduğu gibi nöronlar arasında var olan sinaptik bağlantılar ile yapılmaktadır.

Bu yapay nöron`a perceptron adı verilmektedir. Bir perceptron birçok girdi alır x1, x2,… ve tek bir çıktı üretir:



Şekil 3.6. Perceptron

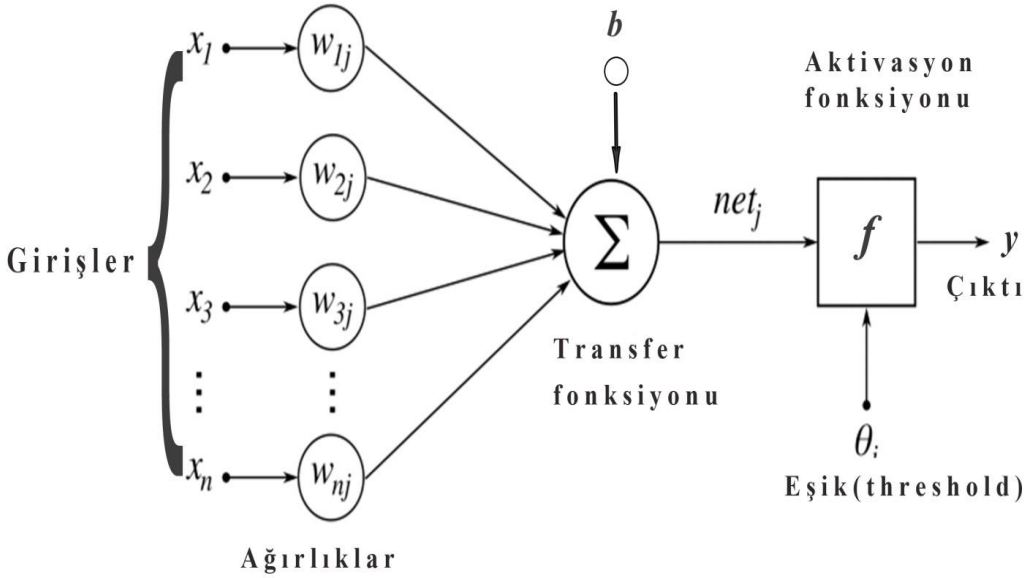
Gösterilen örnekte, perceptron üç girişe sahiptir, x1, x2, x3. Bu girişler farklı uygulamalarda daha fazla veya daha az olabilir. Girdilerin çıktılar üzerindeki etkisini ayarlayabilmek için w1, w2, w3 parametreleri ile belirtilen ağırlık

(weights) değerleri verilmiştir. Ağırlıklar girdi değerleri ile çarpılarak ileri doğru iletilir. Ağdaki en kritik nokta, verilen ağırlıkların eğitim kümesine göre hatayı yayarak olması gereken optimum ağırlık değerlerini hesaplamaktır. Bu ağırlıklar pozitif, negatif veya sıfır olabilir. Ağırlığı sıfır olan nöronların çıktı üzerinde etkisi olmamaktadır. Nöronun çıktısının 0 veya 1 olması $\sum_j w_j x_j$ ağırlıklı toplamın bazı eşik (threshold) değerlerden düşük veya büyük olup olmadığına göre belirlenir. Tıpkı ağırlıklar gibi, eşik (threshold) değeri nöronun bir parametresi olan gerçek bir sayıdır. Daha kesin bir matematiksel terimle ifade etmek için aşağıdaki formül ile yazılabilir:

$$çıktı = \begin{cases} 0, & eğer \ \sum_j w_j x_j \leq \text{eşik (threshold) değeri} \\ 1, & eğer \ \sum_j w_j x_j \geq \text{eşik (threshold) değeri} \end{cases}$$

Eğitim kümesinin öznitelikleri yapay sinir ağına girdi olarak verilerek üretilecek çıktıların gerçek etiket değerlerine yaklaşması yani ağırlıkların en iyi değerlerinin bulunması için güncellenmeleri sağlanır. Eğitimin amacı, ağa gösterilen örnekler için doğru çıktıları üretecek ağırlık değerlerini bulmaktır. İşte bu aşamalarda ki geçen süreye öğrenme adı verilir.

Yapay sinir ağlarının yapı blokları ve işlevi olan nöronların yapısı aşağıdaki gibidir.



Şekil 3.7. Yapa sinir ağı

Transfer fonksiyonu: Bu fonksiyon bir nörona gelen net girdiyi hesaplar. Bunun için değişik fonksiyonlar kullanılmaktadır. En yaygın olan ise ağırlıkların girdiler ile çarpımının toplamıdır.

$$net = \sum w \times x$$

YSA`nın yapı blokunda önemli bir noktada aktivasyon fonksiyonudur. Hücreye gelen net değeri işleyerek nöronun bu girdiye karşılık aktivasyon çıktısını üretir. Aktivasyon fonksiyonunun probleme göre uygun seçilmesi ağın performansını ve başarı oranını önemli derecede etkiler. Hatayı geri yayarken her katmanda türev işlemi yapılır. Dolayısıyla seçilecek aktivasyon fonksiyonlarının türevlenebilir ve türevlerinin kolay alınabilen fonksiyonlardan olması gerekmektedir. Aksi takdirde eğitim süresi uzayacaktır. Günümüzde en yaygın olarak kullanılan aktivasyon fonksiyonu olarak “Sigmoid fonksiyonu” kullanılır.

$$f(x) = \frac{1}{1 + e^{-net}}$$

Sigmoid aktivasyon fonksiyonu sürekli ve türevi alınabilir bir fonksiyondur. Doğrusal olmayışı dolayısıyla yapay sinir ağı uygulamalarında en sık kullanılan fonksiyondur. Bu fonksiyon girdi değerlerinin her biri için 0 ile 1 arasında bir değer üretir.

Bu net değer aktivasyon fonksiyonundan geçirilerek çıkış (out) değeri oluşturulur.

$$y = f(net)$$

3.2.Sözcük Tabanlı Yaklaşım

Duygu analizi yapılandırılmamış verilerde ifade edilen fikirleri, görüşleri ve duyguları analiz etmeyi amaçlayan doğal dil işleme (NLP) görevi olarak görülebilir. Bir metnin veya cümlenin anlamsal yönelimine veya kutupluluğuna (pozitif, negatif veya nötr) göre sınıflandırılması makine öğrenmesi, sözcüğe dayalı yöntemler veya hibrid yöntemler ile yapılabilir.

Sözcüğe dayalı yaklaşım adından da anlaşılacağı gibi görüş kelime sözlüğü veya görüş cümle sözlükleri gibi duygu analizi sürecinin ana kaynakları kullanılmaktadır. Görüş kelimeleri birçok duyarlılık sınıflandırma görevinde bulunmaktadır. Olumlu görüş kelimeleri bazı istenen durumları ifade etmek için kullanılırken, olumsuz görüş kelimeleri bazı istenmeyen durumları ifade etmek için kullanılır. Bu kelimelerinin derlenip duygu sözlüğü oluşturulması için üç temel yaklaşım vardır.

- ✓ Manuel yaklaşım
- ✓ Sözlük tabanlı yaklaşım
- ✓ Korpus tabanlı yaklaşım

Manuel yaklaşım, bir kişi veya gönüllü gruplar tarafından kelimelerin toplanıp ardından kelimelerin ifade ettiği duyguların manuel olarak

etiketlenmesidir. Bazı araştırmacılar bu yaklaşımı geçmişte seçmişlerdir. Ancak manuel yaklaşım çok zaman alıcıdır ve nadiren kullanılır. Bu nedenle otomatik yöntemlerden kaynaklanan hataları önlemek için otomatik yaklaşımlarla birlikte kontrol için kullanılır (Albayrak, 2018). Diğer iki otomatik yaklaşım aşağıdaki alt bölümlerde sunulmuştur.

3.2.1. Sözlük tabanlı yaklaşım

Otomatik yöntemlerden ilki olan sözlüğe dayalı yöntemler çoğunlukla ortak görüş terimleri kümeleri için toplanan açıklamalı sözlüklere dayanır. Bu yaklaşıma güzel bir örnek SentiWordNet'tir. Her ne kadar duyarlılık analizi için güvenilir bir çözüm olsa da, farklı alanlarla uğraşırken veya içerik analiz ederken bazı sınırlamalar vardır.

Sözlük tabanlı yöntemler yalnızca dil bilgisine dayanır, belirli alanlar ve metinler arasında daha güçlüdür. Bununla birlikte, yüksek doğruluk elde etmek zordur. Temel olarak olumlu olumsuz kelimelerden oluşan bir duyarlılık sözlüğü kullanılır. Bir duyarlılık sözlüğüne ait olan sözcükler duygusal sözcükler olarak adlandırılır (Avanço and Nunes, 2014).

Strateji olarak önce manuel küçük bir görüş kelimeler kümesi oluşturuluyor ve daha sonra bu kelimelerin sinonim yani eşanlamlı sözcükleri ile karşıt(zıt) anlamlı sözcükler için WordNet veya başka kaynaklar kullanılarak yeni kelimeler aranır, bulunan yeni sözler bu kümeye eklenir ardından bir sonraki yinelemeli işlem başlar. Yinelenen süreç, başka kelime bulunmadığında durur. İşlem tamamlandıktan sonra, hataları gidermek veya düzeltmek için manuel inceleme yapılabilir.

Sözlüğe dayalı yaklaşım ve ondan toplanan görüş sözcükleri büyük bir eksikliğe sahiptir. Bu yaklaşım etki alanına ve içeriğe özgü yönelimleri olan fikir sözcüklerini bulamama gibi büyük bir dezavantaja sahiptir. Örneğin, bir hoparlör için eğer “sessizse” genellikle olumsuz. Ancak, bir araba için eğer “sessizse” kelimesi olumlu olarak kullanılmaktadır. Korpus temelli yaklaşım bu sorunun üstesinden gelmeye yardımcı olabilir (Liu and Zhang, 2012).

3.2.2. Korpus tabanlı yaklaşım

Otomatik yöntemlerden ikincisi olan korpus temelli yaklaşım da sözlüklere dayanmaktadır. Korpus temelli yaklaşım, içeriğe özgü yönelimleri olan görüş kelimeleri bulma problemini çözmede yardımcı olur. Bahsedilen sözlükleri üretmek için istatistiksel ve semantik teknikler sıklıkla kullanılır. Korpus temelli yaklaşımda yöntemler büyük bir korpus içindeki diğer görüş kelimelerini bulmak için sözdizimsel veya ortak oluşum desenlerine ve ayrıca görüş kelimelerinin bir çekirdek listesine dayanmaktadır. Bu alandaki önemli gelişme, duygu kesinliği kavramını tanıtan Hatzivassiloglou ve McKeown (1997) çalışmasıdır. Çalışmalarına duygu sıfatlarından oluşan bir çekirdek kümesiyle başlamışlardır ve bunlara ek sıfat duygu kelimelerini ve yönelimlerini belirlemek için "ve, veya, ama, ne de…" gibi bağlaçlar için bir dizi dil kuralları kullanmışlardır. Kısıtlamalardan biri birleşik sıfatların genellikle aynı yönelime sahip olduğunu söyleyen "ve" birleşimi ile ilgilidir. Örneğin "Bu araba güzel ve ferahdır" cümlesinde "güzel" in olumlu olduğu biliniyorsa, "ferah" ın da olumlu olduğu sonucuna varılabilir. İnsanlar genellikle "ve" bağlantısının her iki tarafında da aynı görüşü ifade ederler. Buna duygu kesinliği deniliyor.

Ancak, tüm görüş kelimelerini tanımlamak için korpus temelli yaklaşımı kullanmak, sözlük temelli yaklaşım kadar etkili değildir, çünkü tüm kelimeleri kapsayacak şekilde büyük bir korpus hazırlamak zordur (Medhat et al, 2014).

4. VERİ TOPLAMA VE ÖN İŞLEME YÖNTEMLERİ

4.1.Twitter

Twitter, kullanıcıların “tweet” olarak bilinen mesajları gönderdiği ve bunlarla etkileşime girdiği bir çevrimiçi haber ve sosyal ağ hizmetidir. Tweetler başlangıçta 140 karakterle sınırlandırılmış, ancak 7 Kasım 2017'de bu sınır, Çince, Japonca ve Korece dışındaki tüm diller için iki katına çıkarıldı. Kayıtlı kullanıcılar tweet gönderebilir, beğenebilir ve retweetleyebilir, ancak kayıtsız kullanıcılar yalnızca bunları okuyabilir (Rosen, 2017).

Twitter kayıtlı kullanıcıların “tweet” adı verilen kısa mesajları okuma ve yazmasını sağlayan sosyal ağ ve mikro blok hizmetidir. Twitter mesajları 280 karakterle sınırlıdır, kullanıcılar fotoğraf veya kısa video yükleyebilir. Kullanıcılar tweetlerini herkese açık bir şekilde kendi sayfalarında paylaşabilir veya diğer kullanıcılara doğrudan mesaj olarak gönderebilir. Twitter dünya çapında en popüler sosyal ağlardan biridir (Jain and Katkar, 2015).

2018 yılının üçüncü çeyreğinden itibaren, mikro blok hizmeti, aylık 326 milyon aktif kullanıcıya ulaşmıştır. İnsanlar günde 500 milyondan fazla sisteme tweet gönderiyor (Twitter Statistics).

Twitter, kullanıcılarının tweet adı verilen gerçek zamanlı mesajlar göndermelerini sağlayan bir sosyal ağ ve mikro blok hizmetidir. Tweetler yeni zorluklar doğuran ve diğer alanlara göre üzerinde duyarlılık analizleri yapmanın yollarını oluşturan birçok benzersiz özelliğe sahiptir. Aşağıda tweetlerin bazı temel özellikleri verilmiştir (Kumar and Sebastian, 2012):

- ✓ **Mesaj uzunluğu**: Bir Twitter mesajının maksimum uzunluğu 280 karakterdir. Bu, ürün ve film incelemeleri gibi daha uzun metinleri sınıflandırmaya odaklanan diğer duyarlılık sınıflandırma araştırmalarından farklıdır.
- ✓ **Yazma tekniği**: Yanlış yazımların ve siber argoların tweet'lerde ortaya çıkması diğer alanlara göre daha sık görülür. Mesajlar hızlı ve

kısa yazıldığı için, insanlar kısaltmalar, yanlış yazım, ifadeler ve özel anlamlar taşıyan diğer karakterler kullanırlar.

- ✓ **Kullanılabilirlik:** Mevcut veri miktarı çok büyüktür. Facebook ile karşılaştırıldığında, daha fazla kişinin tweet'lerini herkese açık şekilde yayınladığı görülmektedir. Bu nedenle veriler daha kolay erişilebilir olmaktadır. Twitter API'si veri setinin oluşturulması için tweet`lerin toplanmasını kolaylaştırır.
- ✓ **Konular:** Twitter kullanıcıları, belirli bir konu için tasarlanmış diğer sitelerin aksine farklı konular, olaylar veya kişiler hakkında mesajlar yayınlar.
- ✓ **Gerçek zaman:** Bloklar, karakteristik olarak daha uzun ve yazma zaman aldığından belirli zaman aralıklarında güncellenir. Öte yandan, tweet`ler 280 harf ile sınırlıdır ve çok sık güncellenmektedir. Bu daha gerçek zamanlı bir his verir ve olaylara ilk tepkileri temsil eder.

Twitter aşağıda listelenen bazı özellikleri ile tanımlanır (Giachanou and Crestani, 2016)

- ✓ **Tweet:** Twitter'da yayınlanan tek bir mesajdır. En fazla 280 karakter olabilen bir tweet'in içeriği, kişisel bilgilerden veya ürünler, olaylar hakkındaki kişisel görüşlerden oluşmaktadır. Bu içerik bağlantılar, haberler, fotoğraflar veya videolar gibi farklılık gösterebilir
- ✓ **Kullanıcı / Kullanıcı Adı (User/Username):** Bir kullanıcının tweet göndermek için bu platforma kaydolması gerekir. Kullanıcı, kayıt sırasında veya daha sonra mesaj göndermek için kullanılacak bir takma ad (kullanıcı adı) seçer.
- ✓ **Bahsetme (Mention):** Başka bir kullanıcıdan bahsetmek için kullanıcı adının önünde @ sembolünü ve ardından kullandıkları belirli kullanıcı adını (@username) kullanırlar.
- ✓ **Cevaplar (Replies):** Bir tweet'deki cevaplar, gönderinin başka bir tweet`in yanıtı olduğunu ve genellikle sohbetler oluşturmak için kullanıldığını belirtmek için kullanılır. Bahsedilenlere benzer şekilde, @ sembolünü ve ardından başvurdukları kullanıcı adını kullanarak

yaratılırlar. Yanıtlar, yanıtı oluşturan kullanıcı adının yanına yerleştirilir.

- ✓ **Takipçi (Follower):** Takipçiler, kullanıcının tweet'lerini ve etkinliklerini takip eden kullanıcıları ifade eder. Diğer kullanıcıları takip etmek için twitter'deki diğer kullanıcılara ulaşmanın ana yoludur. Twitter'deki kullanıcılar izlediklerinden güncellemeler alırlar ve güncellemelerini izleyenlere gönderirler.
- ✓ **Retweet:** Retweetler, yeniden paylaşılan tweet'leri ifade eder. Bir kullanıcı ilginç bir tweet bulduğunda, tekrar tweet`leme işlevini kullanarak tekrar gönderebilir. Retweetleme, bilgiyi yaymak için güçlü bir araç olarak kabul edilir. Paylaşılan tweet değişmeden kalır ve genellikle yazar kısaltması RT ve ardından yazarın kullanıcı adı (RT @ kullanıcı adı) ile işaretlenir. Retweet de kısa bir yorum içerebilir.
- ✓ **Hashtag**: Hashtag'ler, bir tweet`in belirli bir konuyla ilgi düzeyini belirtmek için kullanılır. # Karakterini ve ardından konu adını (#topic) kullanarak oluşturulan etiketler, gönderilen iletilerdeki bilgileri etiketleme gereksiniminden ortaya çıkmıştır. Etiketler kullanıcılar tarafından kendiliğinden üretilir ve tüm tweet'leri aynı hashtag ile almak için kullanılabilir. Çok sayıda tweet`de görünen Hashtag'ler, trend konuları olarak nitelendirilir.
- ✓ **Gizlilik (Privacy):** Twitter kullanıcının tweet`lerinin herkes tarafından mı yoksa yalnızca onaylı twitter takipçileri tarafından mı görülebileceğine karar vermesini sağlar.

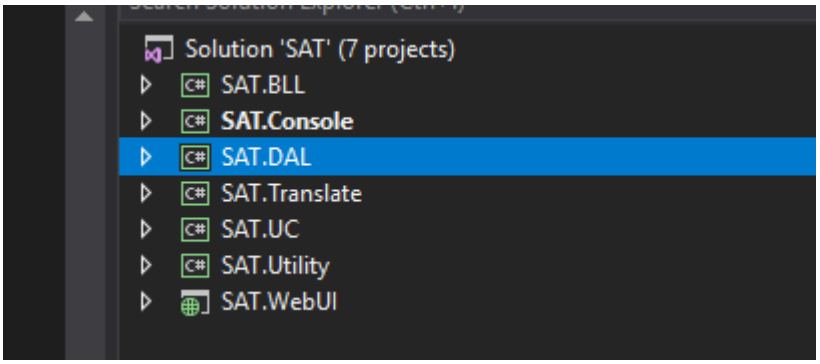
4.2.Veri Setinin Oluşturulması

Büyük veri analizlerine atılan ilk adım, verileri toplamaktır. Bu "veri madenciliği" olarak bilinir. Veri toplama, ilk bakışta göründüğü kadar basit değildir. Veriler her yerden gelebilir. Bu tez çalışması kapsamında yapılan görüş polaritesi sınıflandırma çalışmalarında kullanılan veri seti, Twitter sosyal ağından elde edilmiştir. Twitter bir altın veri madenidir. Diğer sosyal platformlardan farklı olarak, hemen hemen her kullanıcının tweet'leri tamamen halka açık ve etkileyicidir. Analitik çalışma yapmak için büyük miktarlarda veri gerekiyorsa twitter bu konuda çok büyük önem kazanmaktadır. Ayrıca twitter verileri diğer

sosyal medya verileri ile kıyaslandığında oldukça belirgindir. Bu mikroblog hizmetinden veri almak da diğerlerinden daha kolay olduğundan genellikle çalışmalarda twitter sosyal ağı tercih edilmektedir. Bu tercihin ana sebepleri aşağıda belirtilmiştir (Pak and Paroubek, 2010).

1. Twitter'ın API'si, son yirmi dakika içinde belirli bir konuyla ilgili her tweet'i çekme veya belirli bir kullanıcının retweetlenmemiş tweet'lerini çekme gibi karmaşık sorgular yapmasına olanak sağlamaktadır.
2. Mekansal konum bilgilerini kullanarak belirli bir yerde yaşayan kullanıcıların verilerine kolaylıkla ulaşılmasına imkan tanımaktadır.
3. Bu mikroblog platformu farklı kişiler tarafından farklı konular hakkındaki görüşlerini açıklamak için kullanılır, bu nedenle insanların fikirlerinin değerli bir kaynağıdır.
4. Twitter çok sayıda kısa mesaj içerir ve her gün bu mesajların sayısı büyümektedir. Bundan dolayı toplanan korpus isteğe bağlı olarak büyütülebilir.
5. Twitter'ın izleyicileri normal kullanıcılardan ünlülere, şirket temsilcilerine, politikacılara ve hatta ülke başkanlarına kadar çeşitlilik göstermektedir. Bu nedenle, farklı sosyal ve ilgi gruplarından olan kullanıcıların yazılı mesajlarını toplamak mümkündür.
6. Twitter'ın kitlesi birçok ülkeden kullanıcılar tarafından temsil edilmektedir. ABD'deki kullanıcılar daha fazla olsa da, farklı ülkelerde bu sosyal ağ hiç azımsanmayacak kadar kullanılmaktadır. Bu nedenle, farklı dillerde veri toplamak mümkündür.

Bu çalışmada tweetlerin elde edilmesi için C# dili ile geliştirilmiş bir uygulama twitter`den verileri gün bazında almaktadır. Uygulama şekil 9`da gözüktüğü gibi 7 katmandan oluşmaktadır.



Şekil 4.1. SAT uygulaması

Uygulamanın SAT. BLL katmanında yapılan proje Twitter`e bağlantı kurup verileri almak için geliştirilmiştir. Bu projede verileri çekmek için açık kanyak uygulaması olan Tweetinvi kullanılmaktadır. Tweetinvi; geliştiricilerin twitter ile kolayca ve güvenilir bir şekilde etkileşime girmelerini sağlayan bir .NET C # kütüphanesidir. Tweetinvi github ve nuget üzerinde bulunur. Bu kütüphaneyi projeye dahil etmek için aşağıdaki komutları çalıştırmak gereklidir:

$$Install - Package\ TweetinviAPI$$

Tweetinvi kütüphanesi ile twitter arasında bağlantı kurabilmek için twitter üzerinden bir uygulama oluşturarak kimlik bilgisi erişim parametrelerinin elde edilmesi gerekmektedir. Bu parametreleri elde ettikten sonra herkese açık olan tweetlere erişme olanağı tanınmaktadır. Twitter, çalıştırılan her sorgu için kimlik bilgisi gerektirdiğinden, yapılması gereken ilk iş, kimlik bilgilerini ayarlamaktır. Kimlik bilgileri kullanıcı için erişim simgesi olarak adlandırılan iki anahtar ve uygulama için ayrıca tüketici simgesi olarak adlandırılan iki anahtar içerir.

$$Auth.SetUserCredentials("CONSUMER\_KEY","CONSUMER\_SECRET","ACCESS\_TOKEN","ACCESS\_TOK$$

Kimlik bilgileri tanımlandıktan sonra, Tweetinvi ile yapılan tüm işlemlerde bu kimlik bilgileri kullanılır. Daha sonra Tweetinvi`yi kullanarak veriler twitterden alınarak SAT.UC katmanına gönderiliyor, bu katmanda gelen verilen uygun ön-işlemden geçtikten sonra SAT.DAL katmanına geçiriliyor. Bu katmanda gerekli işlemler yapıldıktan sonra veriler veri tabanına kaydediliyor.

Tezin amacı Azerbaycan dilinde gönderilen twitter mesajlarının duygularını analiz etmek olduğu için tez kapsamında analiz edilecek veri olarak, twitterden Azerbaycan dilinde yazılmış tweetler alınmıştır. Twitter sosyal ağında Azerbaycan dili tanımlı olmadığından bu dilde yazılan tweetlerin dili otomatik olarak Türkce olarak belirtiliyor. Bundan dolayı Azerbaycan dilinde yazılan tweetleri almak için Coğrafi konum bilgileri kullanılmıştır. Başkent Bakuden rastgele alınmış X ve Y koordinatlarından oluşan bir nokta başlangıç nokta olarak kabul edilmiştir. Bu noktadan 300 km yarıçapında yerleşen araziden atılan tweet verileri gün bazında çekilmiştir. Azerbaycan'da twitter kullanıcıları arasında Rus, İngiliz ve Türk dilleri aktif olarak kullanıldığından sorgu oluştururken coğrafi konum kullanmanın yanı sıra dil parametresi olarak da Türk dili belirtilmiştir. Bundan dolayı Rus ve İngiliz dilinde yazılmış tweetler gelen veri setinden otomatik çıkarılmış olur. Bununla birlikte bu arazide Türkiye Türkçesi ile yazılan tweetleri elemek için tweet yazan kullanıcıların profillerinde kullandıkları konum bilgileri kontrol edilmekte ve bu konumlar Azerbaycan ve Azerbaycan'ın illeri dışında bir yer içeriyorsa bu tweetlerde veri setinden çıkartılmaktadır. Tweetinvi arama metodu "MaximumNumberOfResults" parametresi içerir ve bu parametre twitterden gelecek tweetlerin maksimum sayısını belirler.

```
BussinesLogicOperation blogic = new BussinesLogicOperation();
phdSearch psearch = new phdSearch();
psearch.latitude = 40.474644;
psearch.longitude = 49.913266;

psearch.radius = 300;
psearch.maximumNumberOfResults = 100000;

psearch.lang = "Turkish";
psearch.since = DateTime.Now.AddHours(-24);
psearch.until = DateTime.Now;

blogic.prepareTweetsForDB(psearch);
```

Şekil 4.2. Sorgu parametrelerin belirlenmesi

4.3.Metin Sınıflandırması İçin Özellik Seçimi

Herhangi bir sınıflandırma işleminden önce, yapılması gereken en temel görevlerinden biri özellik seçimidir. Diğer sınıflandırma işlemlerinde özellik seçimi kullanılsa da, metin sınıflandırmada özelliklerinin yüksek boyut ve gereksiz (gürültülü) özelliklerin varlığı nedeniyle bu tarz sınıflandırmada özellik seçimi önemlidir. Metin verilerinden öğrenme yaparken en önemli problem her bir dokümanın farklı uzunlukta olmasıdır. Böyle durumlarda her bir kelimeyi bir özellik olarak ele almak gerekir. Oysa ki bu durumu uygulamak büyük çaplı verilerde oldukça zordur. Bu durumun üstesinden gelebilmek için en önemli yöntemlerden birisi ise Kelime Çantası Tekniği (Bag of Words) (BOW)'dir. Bu yöntemde her bir kelime benzersiz olarak ele alınır ve her bir benzersiz kelimenin, doküman içerisinde tekrar etme sayısı bulunur. Sınıflandırmada öğeler özellikleriyle temsil edilir. Bizim durumumuzda, tweet'ler kelimeleri ile temsil edilir, bu yüzden kelimeler özellikler olarak kullanılmaktadır.

Makine öğrenmesinde metin verilerini belirli bir makine tarafından okunabilir formata dönüştürmek için temel bir sürece ihtiyaç vardır. Bu sürece özellik çıkarma denir ve özellik çıkarma için yukarıda anlatılan BOW yöntemi kullanılmıştır. Bu örnekte ilk olarak tweet'lerdeki benzersiz kelimelerin listesi çıkarılır. Daha sonra bu kelimelerin tweetler içerisindeki tekrar sayıları hesaplanır.

Şekil 4.3. Tweet'lerin vektör gösterimi

Bir sonraki adım, verilen kelimelerin boyutuna göre vektör gösterimi oluşturmaktır. Toplamda 14 adet benzersiz kelime vardır. Bu tüm vektörlerin boyutunun 14 olacağı anlamına gelmektedir. Her bir cümle için vektör gösterimi Şekil 11'deki gibidir Bu vektör gösterime örnek aşağıdakılar verilebiler;

1. Sayı Vektörleri (Count Vectors)
2. TF-IDF Vektörleri (TF-IDF Vectors)
   - ✓ Kelime seviyesi
   - ✓ N-Gram seviyesi

Sayı Vektörleri (Count Vectors): her satırın korpustan bir tweet`i temsil ettiği, her sütun korpustan bir terimi temsil ettiği ve her hücrenin belirli bir tweet`de belirli bir terimin frekans sayısını temsil ettiği veri kümesinin matris gösterimidir.

TF-IDF Vektörleri: TD-IDF puanı bir terimin doküman içerisindeki önemini gösteren istatistiki yöntem ile hesaplanmış ağırlık faktörüdür. TF-IDF puanı iki terimden oluşur:

- **TF:** Terim Frekansı bir terimin belgede ne sıklıkta gerçekleştiğini ölçmek için kullanılmaktadır. Her belgenin uzunluğu farklı olduğu için, bir terimin uzun belgelerde daha fazla kısa belgeler de ise daha az oranda görünmesi mümkündür. Bu nedenle genellikle normalizasyonun bir yolu olarak terim frekansı, belge uzunluğuna (yani, belgedeki toplam terim sayısına) bölünür.

$$TF(t) = (Bir\ belgede\ "t"\ teriminin\ kaç\ kez\ görün\text{düğü})/(Belgedeki\ toplam\ terim\ sayısı)$$

- **IDF:** Ters Belge Sıklığı bir terimin ne kadar önemli olduğunu ölçmek için kullanılmaktadır. TF hesaplanırken tüm terimler eşit derecede önemlidir. Fakat "ile", "ve" gibi bazı terimlerin çoğu zaman ortaya çıkabileceği, ancak çok az öneme sahip oldukları bilinmektedir. Bu nedenle toplam doküman sayısının, terimin geçtiği dokuman sayısına bölümün logaritmasının mutlak değeri alınmaktadır.

$$IDF(t) = \log\_\, e\,(Toplam\ dokuman\ sayısı)\ /\ (\ İçinde\ "t"\ \ terimi\ olan\ belge\ sayısı)$$

4.4.Veri Ön İşleme

Bu bölümde sosyal ağlarda kullanıcılar tarafından oluşturulan yorumlar üzerinde veri analizi yapmak için yapılan ön işlemeler anlatılmıştır. Tweet'lerde kullanılan dilin değişen ve tahmin edilemeyen doğası nedeniyle belirli tweet`lerin

belirteçlerini standartlaştırmak için ön işleme tekniklerinin kullanılması gerekmektedir. Twitter tarafından uygulanan 280 karakter sınırı nedeniyle çoğu tweet'in bir tür gramer veya yazım hatası, kısaltma, argo veya lehçe kelimesi içermesi muhtemeldir. Bu tez çalışmasında uygulanan teknikler, özellikle sosyal medya verileri üzerinde yapılan duyarlılık analizi çalışmalarında yaygın olarak kullanılan yöntemlerdir. Toplanan bu veriler ham veridir. Ham veri, farklı ön işleme adımları kullanarak yapılandırılmış veri biçimine dönüştürmek için kullanılan yapılandırılmamış veri anlamına gelir (Arya and Bhagat). Veriler üzerinde sınıflandırıcı uygulamak için, ham verileri önce işlemek veya temizlemek önemlidir. Ön işlemenin görevi; hashtag ,twitter notasyonları (@, RT) , ifadeler, URL'ler ve durma(etkisiz) kelimelerin kaldırılması , argo sözcüklerinin belirlenmesi ve uzun kelimelerin sıkıştırılmasını içerir. Gerçekleştirilen ön işleme adımlarından bazıları aşağıda açıklanmıştır.

1. İşaretçilerin Tespiti ve Kaldırılması (kullanıcı adları ve hashtagler)
   Twitter, kullanıcıların mesajlarına hashtag (#) ve kullanıcı referansları (@) eklemesini izin verir. Tweet`lerde bir kullanıcı adının önünde "@" işaretçisi kullanarak diğer kullanıcılara işaret edebilir. Ayrıca kullanıcılar bir kategoriye ait tweet'leri "#" kullanarak etiketler. Bu işaretçilerle işaretlenmiş kelime ve karakterleri tweet`lerden kaldırılıyor.
2. URL Tespiti ve Kaldırılması
   Birçok tweet, sınırlı karakter kısıtlamasından dolayı verilebilecek içerikten daha fazla içerik paylaşmak için URL'ler içerir. URL'deki içerik, kullanıcının ifade etmeye çalıştığı duyguya ilişkin ek bilgiler sağlayabilir, ancak URL'leri içeriklerine göre taramak çok pahalı olabilir. Ön işleme sırasında URL `ile ilişki tüm içerikler kaldırıldı.
3. Tekrar Tweetlerin Kaldırılması
   Verilerin tekrar tekrar işlenmemesi ve sınıflandırıcıya verilen eğitim verisinde tekrar eden verilerin engellenmesi için Tweetinvi kütüphanesinde kullanarak alınan tweetleri veri tabanına yazmadan önce " bool IsRetweet { get; }" özelliğin ne döndürdüğü kontrol ediliyor. Eyer "true" dönüyorsa bu tweeti veri setine ilave edilmiyor.
4. Noktalama İşaretlerinin Belirlenmesi ve Kaldırılması

Kullanıcıların mikro blok ortamında, uygun gramerden uzak durmak ve duygularını daha kolay iletmek için aşırı noktalama kullanması yaygındır. Noktalama işaretleri mesajın polaritesine dair bir fikir verebilir. Örneğin, ünlem işaretleri genellikle kutupsal mesajlar olan güçlü vurguyu ifade etmek için kullanılır. Bu adımda gereksiz özelliklerden kaçınmak için tweet`lerden ilgisiz noktalama işaretleri kaldırıldı.

5. Rakamların Kaldırılması

   Sayılar metin verilerinin analizi ile ilgili değildir, bu nedenle toplanan tweet`lerden sayılar kaldırıldı. Ama bu sayılar bağımsız sayı olarak kullanılmış ise kaldırılıyor. Örneğin “Acun Tv8'i satmış anlayın artık nasıl bir krizin içine düştük.” Tweet`indeki 8 kaldırılmıyor. Buna karşılık “Bugün sabah 8 de uyanıp, kahvaltıyı edip hemen ev temizliğine girişmem. Ev hanımı olmuşum ya ben” tweet`inde kullanılan “8” sayısı kaldırılıyor.

6. Karakter değiştirmek

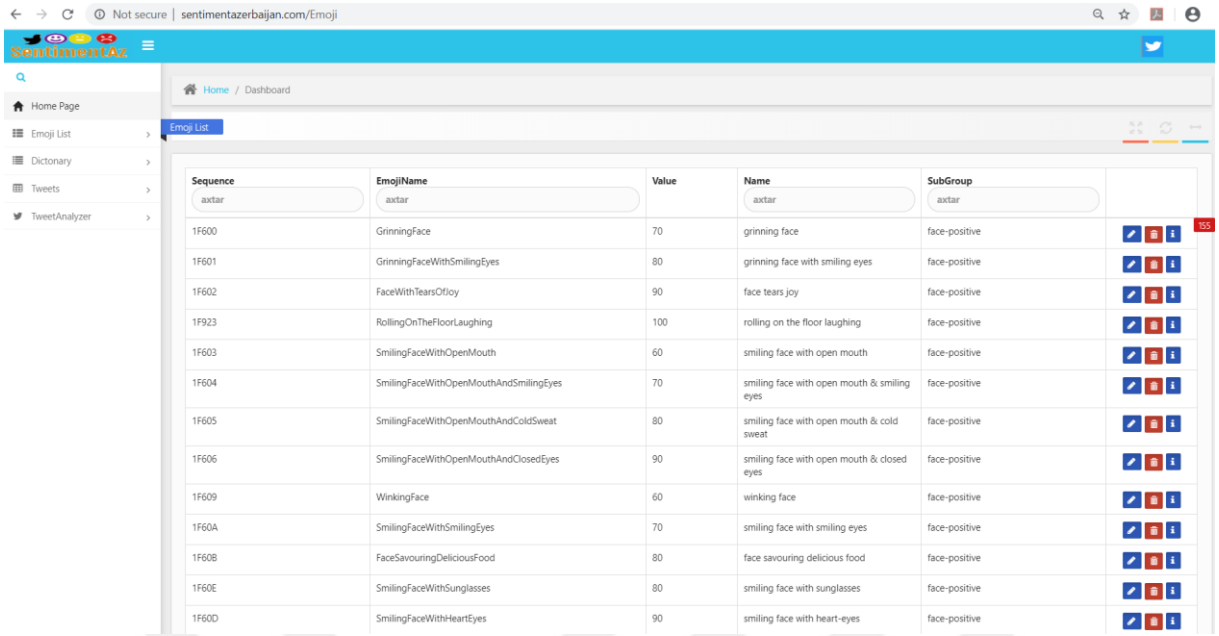
   Kullanıcılar tweet`lerde, metin parçasına olumlu veya olumsuz anlam katan bazı karakter birleşimleri kullanmaktadırlar. Bu karakter birleşimleri veya ifadeler kullanıcılar tarafından tweet`lerde duyguları özlü bir şekilde ifade etmenin kolay bir yolu olarak kullanılır. Karakter birleşimleri oluşturulan kod karşılığı ile değiştirilmektedir.

Çizelge 4.1: Karakter değiştirmek

Olumlu		Olumsuz	
Karekter birleşimi	Kod karşılığı	Karekter birleşimi	Kod karşılığı
:d :D, ;d ;D, =d =D, xd XD, :p :P	LLW01d, LLW02d, LLW03d, LLW04d, LLW01p	:o :O, ;o ;O	LLW01o, LLW02o
:), ;), =),)),))), :-), ;-), :)), ;)), ^-^	SHF01X, SHF02X, SHF03X, SHF0XX, SHFXXX, SHF14X, SHF24X, SHF1XX, SHF2XX, SHF545	:(, ;(, =(, ((, (((, :-(, ;-), :((, ;((,	SHF01Y, SHF02Y, SHF03Y SHF0YY, SHFYYY, SHF14Y, SHF24Y, SHF1YY, SHF2YY

7. Emoji Çevirme

   Emojiler: noktalama işaretleri ve alfa numerik karakterler kullanarak yüz ifadelerini temsil eden simgelerdir. Tweet`lerdeki duygu içeren emojiler unicode karşılığı ile değiştirilmiştir. Bu emojilerin Unicode listesi veri tabanında bir tabloda tutulmaktadır. Alınan tweet karakter kontrol edilip kullanılan "emoji"`ler unicode karşılığı ile değiştirilmiştir. Örneğin "Buraya daha iyi yakıştı😎" tweet`i analiz aşamasına gelmeden önce "buraya daha iyi yakıştı 1f60e" bu formaya getirilmiştir. Emojiler farklı bir sütunda tutulmuştur.



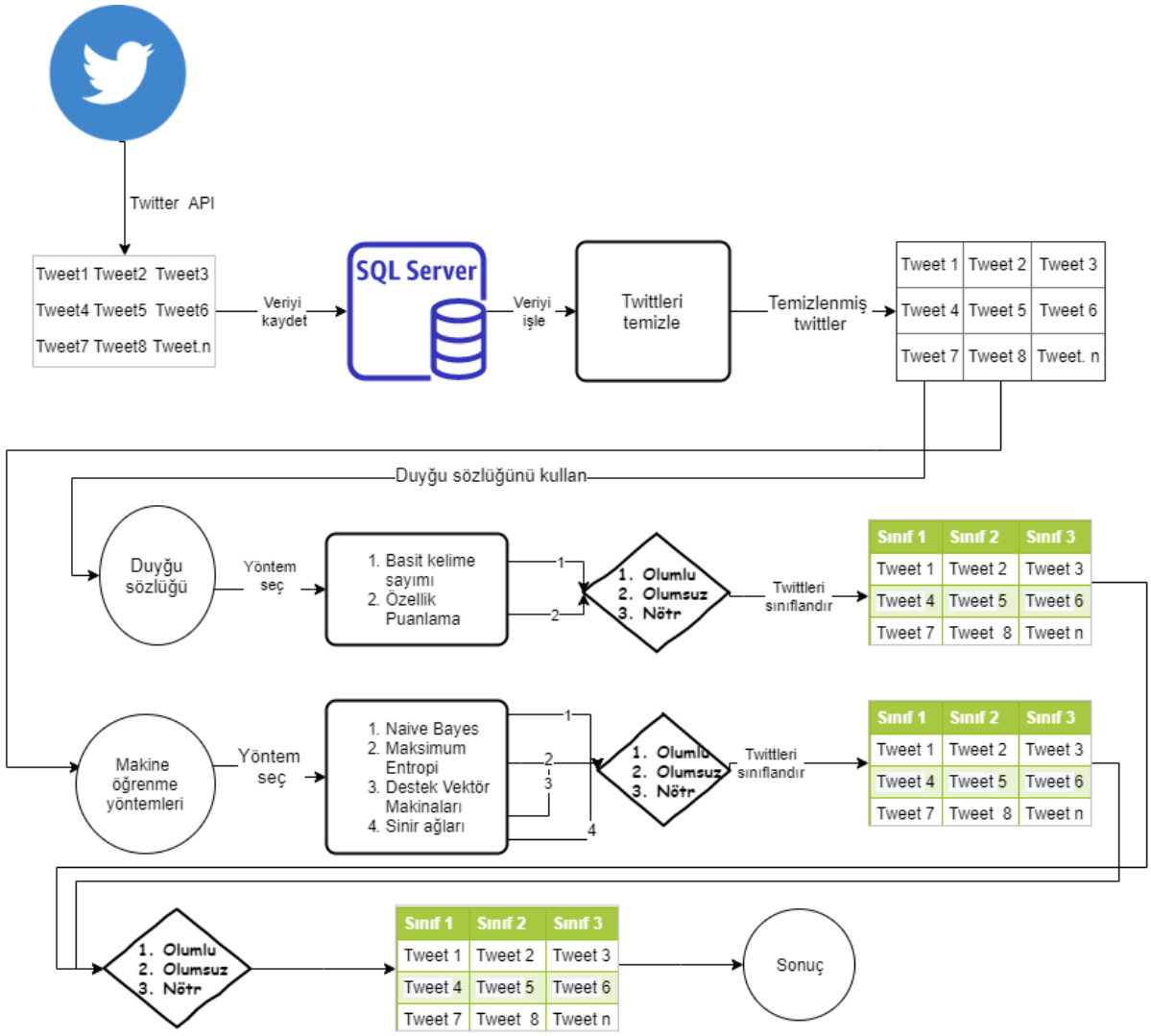
Şekil 4.4. Emoji çevirme

8. Küçük harfe çevirme

   Tweet`ler en son olarak hepsi küçük harfe çevriliyor. Çevrilme zamanı Türkçe karakterlerin uygun karşılıkları yazılıyor.

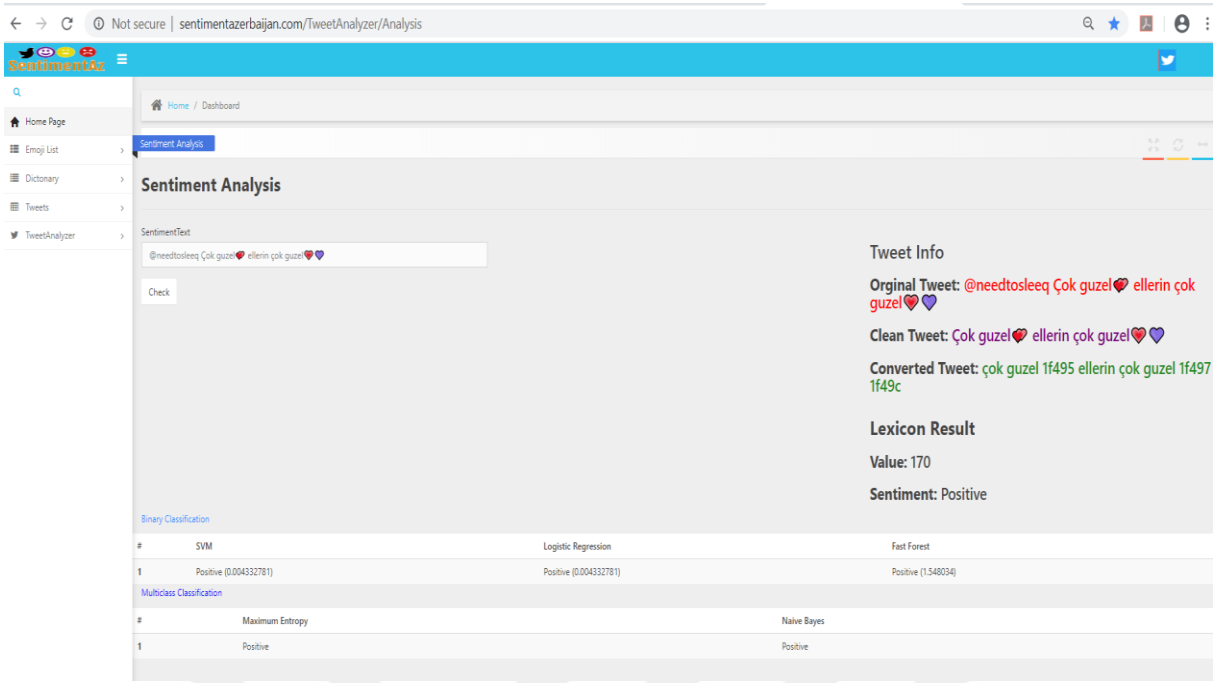
5. AZERBAYCAN DİLİ İÇİN DUYARLILLK ANALİZ TEKNİKLERİ

Tez kapsamında Azerbaycan dili için geliştirilen “SentimentAz” projesinde makine öğrenme yöntemleri ve sözlük tabanlı yöntemleri birlikte kullanarak sistem geliştirilmiştir. Sistemin çalışması yansıtan diyagram aşağıda verilmiştir.



Şekil 5.1. Sistemin ana akış diyagramı

Geliştirilen sistem web tabanlı bir sistem olduğundan her kesin kullanmasına açık bir şekilde web üzerinden bu URL http://sentimentazerbaijan.com/ adresinden yayınlanmaktadır.



Şekil 5.2. Duygu analizi online sistemi

5.1.Duygu Sözlüğünün Oluşturulması

Azerbaycan dili için duygu sözlüğünü oluşturmak için farklı yöntemler kullanılmıştır. Bu yöntemler bazıları aşağıdakilerdir.

- SentiWordNet

Azerbaycan dili için yeterli duygu sözcüklerini içeren sözlük bulunmadığından "SentiWordNet_3.0.0_20130122" kütüphanesini web üzerinden indirip burada bulunan tüm kelimeleri Google firmasının sunduğu "Cloud Translation API-> translate.googleapis.com" servisini kullanarak bu sözlükde bulunan kelimeler Azerbaycan ve Türk dillerine çevrilerek yeni duygu kelimelerini içeren sözlük yaratılmıştır.

5.2.Sözlük Tabanlı Yöntemler

Sözlüğe dayalı yöntemlerin temel varsayımı, bir belgenin duyarlılığının bu belgede kullanılan baskın duyarlılık sözleriyle belirlenmesidir. Bu yaklaşımlardan "Kelime sayımı" ve "Özellik puanlama" yöntemleri kullanılmıştır.

5.2.1. Kelime sayımı

Paylaşılan tweet`ler bir veya daha fazla kelime ve karakter birleşimlerinden oluşmaktadır. Bu tweet`lerin "kelime sayım" yöntemi ile duygu polaritesini belirlemek için bazı tanımlamalar yapılmıştır. Azerbaycan dili için oluşturulan duygu sözlüğü s, verilen tweet t ve bu tweet`in içerdiği kelimeler ise k ile belirtilmiştir. Burada $t = \{k_1, k_2, k_3, k_4, k_n\}$ ile ifade edilmektedir ayrıca $k_\iota\{1 \leq i \leq n\}$ olacak şekilde tanımlanmıştır. Verilen $t$ tweet`in içerdiği olumlu kelimelerin sayısını $olumlu(s,t)$, olumsuz kelimelerin sayısını ise $olumsuz(s,t)$ ile gösterilmektedir. Bu tweet`in içerdiği duygu sözcüklerinin toplam sayısı ise $toplam(s,t)$ ile gösterilmektedir.

$$toplam(s,t) = olumlu(s,t) - olumsuz(s,t) \quad (5.1)$$

Sonuç olarak bir tweet`in duyarlık polaritesini aşağıdaki formül ile belirleniyor:

$$KS(s,t) = \begin{cases} 1, & toplam(s,t) > 0 \\ -1, & toplam(s,t) < 0 \\ 0, & toplam(s,t) = 0 \end{cases} \quad (5.2)$$

Burada $toplam(s,t) > 0$ çıkarsa tweet`in polaritesi olumlu, $toplam(s,t) < 0$ çıkarsa tweet`in polaritesi olumsuz veya $toplam(s,t) = 0$ ise tweet`in polaritesi nötr olarak kabul edilmektedir.

5.2.2. Özellik puanlama

"Özellik puanlama" sözlük yaklaşımı kullanarak bir tweet`in duyarlılık polaritesini hesaplamak için, verilen tweet'in içerdiği her bir kelime, duygu sözlüğü içinde yer alan sözcüklerle karşılaştırılmalıdır. Kelimeler duygu sözlüğündeki kelimler ile eşleşirse, bu kelimelerin karşılığı olan sayısal değerler alınarak toplanır. Bu işlemlerin sonunda tweet başına puan belirlenir, bu da tweet'lerin sayısal olarak diğer tweet'lerle karşılaştırılmasını sağlar. Bu yöntem ile duygu polaritesi belirlemek için aşağıdaki tanımlamalar yapılmıştır. Kelime sayım yönteminde kullanıldığı gibi bu yöntemde kullanılan duygu sözlüğü s verilen tweet ise t ile belirtilmiştir. Diğer yöntemden farklı olarak bu yöntemde tweet`in

içerdiği duygu kelimelerin sayımı değil bu kelimelere verilen puanlar kullanılır Bu puanları belirtmek için $t = \{f_1, f_2, f_3, f_4, f_n\}$ ve $f_i\{1 \leq n \leq n\}$ tanımlanması yapılmıştır.

Verilen bir tweet`in sonuç puanını hesaplamak için aşağıdaki formül kullanılmıştır:

$$toplam(s,t) = \sum_{i=1}^{n} puan(s, f_i) \quad (5.3)$$

Nihai sonucu kullanarak karar vermek için bir eşik değer seçilir. Bu yöntemde eşik değer olarak $\delta = 0$ seçilmiştir.

$$PS(s,t) = \begin{cases} 1, & toplam(s,t) > \delta \\ -1, & toplam(s,t) < \delta \\ 0, & toplam(s,t) = \delta \end{cases} \quad (5.4)$$

6. DENEY ÇALIŞMALARI VE BULGULAR

Bu bölümde tez çalışmasında tasarlanan sürecin deneysel değerlendirmelerine ve bulgularına yer verilmiştir. Deneysel çalışma sonuçlarının ve bulgularının anlaşılabilmesi için öncelikle performans metrikleri ele alınacaktır.

6.1.Değerlendirme

Değerlendirme, çıkan sonuçların güvenilirlik ve iyileştirmesinin ölçüldüğü bir anahtardır. Bir sınıflandırıcıyı değerlendirmenin iki yolu vardır: Çapraz doğrulama ve test verileri. Çapraz Doğrulama hem eğitim hem de test için yeterli veri yoksa veya test için standart bir veri setinin olmadığı durumlarda kullanılır. Böyle bir durumda, test için bir miktar veri tutulur ve kalan eğitim için kullanılır. Ancak bu çalışmada yeterli miktarda eğitim ve test verisi bulunduğundan değerlendirme yöntemi olarak test verileri kullanılmıştır.

6.2.Değerlendirme ölçütleri

Karmaşıklık matrisi, bir sınıflandırma algoritmasının tablo veya matris şeklinde görsel performans değerlendirmesidir. Matrisin her sütunu tahmini sınıflamaları temsil ederken ve her satır gerçek tanımlanmış sınıflandırmaları temsil eder. Bu gösterim bir sınıflandırıcı modelin değerlendirilmesine yardımcı olmak için kullanışlı bir yoldur. Ayrıca karmaşıklık matrisi, gerçek değerleri bilinen bir dizi test verisi ile sınıflandırma modelinin performansını tanımlamak için sıklıkla kullanılan bir tablodur. Bu değerlendirmenin 4 parametresi vardır (Hossin and Sulaiman, 2015).

Çizelge 6.1: Değerlendirme parametresi

		Tahmini		
		Olumlu (+)	Olumsuz (-)	Tarafsız (0)
Gerçek	Olumlu (+)	TP_+	E_{+-}	E_{+0}
	Olumsuz (-)	E_{-+}	TP_-	E_{-0}
	Tarafsız (0)	E_{0+}	E_{0-}	TP_0

Doğru Pozitif (TP): Algoritma evet çıktısı verdi, asıl durum da evet. Örnek: olumlu tweet`leri olumlu sınıf olarak sınıfladığımızda **TP** değeri bulunur.

Doğru Negatif (TN): Algoritma hayır çıktısı verdi, asıl durum da hayır. Örnek: olumsuz tweet`leri olumsuz sınıf olarak sınıfladığımızda **TN** değeri bulunur.

Yanlış Pozitif (FP): Algoritma evet çıktısı verdi ama asıl durum hayır. Burada olumlu bir tweet`i, olumsuz veya tarafsız olarak sınıfladığımızda FP değeri bulunur.

Yanlış Negatif (FN): Algoritma hayır çıktısı verdi ama asıl durum evet. Burada olumsuz bir tweet`i, olumlu veya tarafsız tweet olarak sınıfladığımızda FN değeri bulunur.

Karmaşıklık matrisi sayesinde doğruluk (accuracy), duyarlılık (precision), hassasiyet (recall), F1 skoru gibi metrikler hesaplanabilir.

- **Doğruluk (Accuracy)** Doğruluk, en sezgisel ve basit performans ölçüsüdür. Doğru olarak yapılan tahminlerin tüm tahminlere oranıdır

$$Accuracy = \frac{TP_+ + TP_- + TP_0}{N}$$

- **Duyarlılık (Precision)** Duyarlılık, doğru tahmin edilen pozitif gözlemlerin, tahmin edilen toplam pozitif gözlemlere oranıdır. Buna Pozitif Öngörücü Değer (Positive Predictive Value) de denir.

$$Precision = \frac{TP}{TP + FP}$$

- **Hassasiyet (Recall)** Pozitif durumların ne kadar başarılı tahmin edildiğini gösterir. Sınıflamadaki tüm gözlemlere yönelik olarak doğru tahmin edilen pozitif gözlemlerin oranıdır. Hassasiyet sınıflandırıcıların bütünlüğünün bir ölçüsü olarak düşünülebilir.

$$Precision = \frac{TP}{TP + FN}$$

- **F1 skoru** Hassasiyet ve duyarlılığın harmonik ortalamasıdır. Bu nedenle, hem yanlış pozitif hem de yanlış negatifleri hesaba katar. Özellikle düzensiz bir sınıf dağılımının olduğu durumlarda F1 skoruna bakmak, doğruluk değerine bakmaktan daha yararlıdır. Ayrıca F1-skoru ekstrem durumları cezalandırmak için aritmetik ortalama yerine harmonik ortalamayı kullanıyor

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

6.3. Hesaplama Denemeleri

Makine öğrenme algoritmalari ile yapılan deneyler aşağıdakı çizelgelerde belirtilmişdir.

6.3.1. Naive Bayes denemeleri

Naive Bayes Algoritması 1500, 2500 ve 4500 eğitim veri setleri ile yapılan deneylerde en yüksek %68.56 doğruluk oranı ve %67.53 F1 Skoru elde edilmiştir. Çizelge 6.1'de verilen doğruluk ve F1 Skoru sonuçlarına bakıldığında aşağıdaki bulgulara ulaşılmıştır:

Çizelge 6.2: Naive bayes deneme sonuçları

Veri sayısı	n-gram	Doğruluk	F1 skoru
1500	Unigram	65,87	67,53
	Bigram	65.87	68.35
	Trigram	65.87	68.75
2500	Unigram	63.94	60.55
	Bigram	64.91	62.18
	Trigram	63.94	61.86
4500	Unigram	68.34	59.00
	Bigram	68.56	59.53
	Trigram	67.88	58.77

- Eğitim veri miktarının sonuçlara etkisi incelenmek için 1500, 2500, 4500 sayıda eğitim verisi ile denemeler yapılmıştır. Yapılan denemelerde eğitim veri miktarı arttığında doğruluk oranın artığı gözükmektedir.
- Doğruluk ve F1 Skoru karşılaştırıldığında alınan sonuçların büyük ölçüde örtüştüğü görülmektedir. Bu durum veri dağılımının düzenli olduğunu göstermektedir

6.3.2 Maksimum entropi

Maksimum entropi algoritması 1500, 2500 ve 4500 eğitim veri setleri ile yapılan deneylerde en yüksek %64.52 doğruluk oranı ve %63.41 F1 Skoru elde edilmiştir. Çizelge 6.1'de verilen doğruluk ve F1 Skoru sonuçlarına bakıldığında aşağıdaki bulgulara ulaşılmıştır:

Çizelge 6.3: Maksimum entropi deneme sonuçları

Veri sayısı	n-gram	Doğruluk	F1 skoru
1500	Unigram	58,65	58,53
	Bigram	59.87	58.35
	Trigram	58.87	68.75
2500	Unigram	60.94	60.55
	Bigram	61.91	61.18
	Trigram	59.94	59.86
4500	Unigram	64.52	63.41
	Bigram	63.41	62.53
	Trigram	62.44	62.77

- Eğitim veri miktarının sonuçlara etkisi incelenmek için 1500, 2500, 4500 sayıda eğitim verisi ile denemeler yapılmıştır. Yapılan denemelerde eğitim veri miktarı artığında doğruluk oranın artığı gözükmektedir.
- Doğruluk ve F1 Skoru karşılaştırıldığında alınan sonuçların büyük ölçüde örtüştüğü görülmektedir. Bu durum veri dağılımının düzenli olduğunu göstermektedir

6.3.3 Destek vektör makinaları

Destek vektör makinaları algoritması 1500, 2500 ve 4500 eğitim veri setleri ile yapılan deneylerde en yüksek %74.28 doğruluk oranı ve %73.38 F1 Skoru elde

edilmiştir. Çizelge 6.1'de verilen doğruluk ve F1 Skoru sonuçlarına bakıldığında aşağıdaki bulgulara ulaşılmıştır:

Çizelge 6.4: Destek vektör makinaları deneme sonuçları

Veri sayısı	n-gram	Doğruluk	F1 skoru
1500	Unigram	72,02	70,53
	Bigram	71.45	72.62
	Trigram	70.41	70.65
2500	Unigram	73.94	72.83
	Bigram	72.78	70.32
	Trigram	72.45	71.43
4500	Unigram	74.28	73.38
	Bigram	72.48	72.01
	Trigram	72.30	71.62

- Eğitim veri miktarının sonuçlara etkisi incelenmek için 1500, 2500, 4500 sayıda eğitim verisi ile denemeler yapılmıştır. Yapılan denemelerde eğitim veri miktarı artığında doğruluk oranın artığı gözükmektedir.
- Doğruluk ve F1 Skoru karşılaştırıldığında alınan sonuçların büyük ölçüde örtüştüğü görülmektedir. Bu durum veri dağılımının düzenli olduğunu göstermektedir

7. SONUÇ

Günümüzde duygu analizi birçok alanda kullanılan çeşitli uygulamalara sahip olan bir veri analiz yöntemidir. Duygu analizi konusunda yapılan çalışmalarda elde edilen bilgi birçok alanda önemli bir kazanç sağlamaktadır. Bu yöntem farklı dillerde pek çok araştırmacı tarafından ele alınıp geliştirilmeye çalışılsa da Azerbaycan dili için yapılan çalışmalar çok kısıtlı kalmıştır. Azerbaycan dili için yapılan bu çalışmanın araştırmacıları yeni ve verimli çalışmalar yapmaya imkan sağlayacağı düşünülmektedir.

Bu tez çalışmasının temel amacı Azerbaycan dilinde yazılmış tweet`lerin duygu polaritesinin belirlenmesini sağlamaktır. Tweeter verilerine dayanarak cümleyi sınıflandırmak için bir dizi makine öğrenme teknikleri ve sözcük tabanlı analiz teknikleri uygulanmıştır. Çalışmada kullanılmış veri setleri çeşitli önişlemlerden geçirilerek makine öğrenmesi teknikleri ve sözlük tabanlı yaklaşımlarda kullanılmaya hazır hale getirilmiştir. Veri setleri hazırlanırken Kelime Torbası Tekniği'nden (BOW) yararlanılarak 2 tip özellik vektörü kullanılmıştır.

Çalışmadaki bir diğer amaç çeşitli geleneksel sınıflandırma teknikleri ve onların doğruluk oranlarının karşılaştırılmasıdır. Çalışma kapsamında ele alınan makine öğrenmesi yöntemleri "Naive Bayes, Maksimum Entropi ve Destek Vektör Makineleri olarak gruplandırılmıştır. Veri seti üzerinde yapılan deneme çalışmalarında SVM yönteminin diğerlerine kıyasla daha iyi sonuç verdiği belirlenmiştir. Tüm deneyler Visual studio 2019 aracı kullanılarak "Microsoft.ML" kütüphanesini kullanarak geliştirilen program ile gerçekleştirilmiştir.

Ayrıca tez kapsamında duygu analizi yapmakla birlikte Azerbaycan dili için duygu sözlüğü ve Stop Words (Etkisiz Kelimeler) listesi oluşturmak için gerekli çalışmalar yapılmıştır.

KAYNAKLAR DİZİNİ

Appel, O., Chiclana, F. and Carter, J., 2015, Main Concepts, State of the Art and Future Research Questions in Sentiment Analysis. Acta Polytechnica Hungarica Vol. 12, No. 3,

Albayrak. A., 2018, Duygu Analizinde Farklı Vektör Temsil Yöntemleri Ve Sınıflayıcıların Karşılaştırılması

Ayan, S. ve Erdoğmuş, Ş., 2014, Destek Vektör Makineleriyle Sınıflandırma Problemlerinin Çözümü İçin Çekirdek Fonksiyonu Seçimi, Eskişehir Osmangazi Üniverssitesi İİBF Dergisi Nisan 2014, 9(1), 175- 198

Avanço, L.V and Nunes, M. G. V., 2014, Lexicon-based Sentiment Analysis for Reviews of Products in Brazilian Portuguese Brazilian Conference on Intelligent Systems

Arya, P. and Bhagat, A., 2018, Deep Survey on Sentiment Analysis and Opinion Mining on Social Networking Sites and E-Commerce Website *International Journal of Engineering Science and Computing, March 2017*

Can, U. və Alatas,B., 2017, Duygu Analizi ve Fikir Madenciliği Algoritmalarının İncelenmesi Int. J. Pure Appl. Sci. 3(1): 75-111 (+)

Cambria, E., Schuller, B., Xia, Y. and Havasi, C., 2013, New Avenues in Opinion Mining and Sentiment Analysis. IEEE Intelligent Systems, 28(2):15-21

Chory, R.N., Nasrun, M. and Setianingsih, C., 2018, Sentiment Analysis on User Satisfaction Level of Mobile Data Services Using Support Vector Machine (SVM) Algorithm. IEEE International Conference on Internet of Things and Intelligence System (IOTAIS)

Desai, M and Mehta, M.A., 2016, Techniques for Sentiment Analysis of Twitter Data: A Comprehensive Survey International Conference on Computing, Communication and Automation

Duwairi, R. M., Marji, R., Sha'ban, N. and Rushaidat, S., 2014, Sentiment Analysis in Arabic Tweets. 5th International Conference on Information and Communication Systems (ICICS)

KAYNAKLAR DİZİNİ (devam)

Eliaçık, A. B. ve Erdoğan, N., 2015, Mikro Bloglardaki Finans Toplulukları için Kullanıcı Ağırlıklandırılmış Duygu Analizi Yöntemi Bilgisayar Mühendisliği Bölümü, İstanbul Teknik Üniversitesi,İstanbul

Farooq, U., Dhamala, T.P.,Ouzrout, Y and Qadir, M.A., 2015, A Word Sense Disambiguation Method for Feature Level Sentiment Analysis 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)

Go, A., Bhayani, R., and Huang,L., 2009, Twitter Sentiment Classification using Distant Supervision. CS224N Project Report, Stanford. Vol.1, No.12.

Giachanou. A. and Crestani. F., 2016, A survey of Twitter sentiment analysis methods. ACM Comput. Surv. 49, 2, Article 28, 41 pages.

Hu, M. and Liu B, 2004, Mining and Summarizing Customer Reviews. Department of Computer Science University of Illinois at Chicago 851 South Morgan Street Chicago, IL 60607-7053

Hossin, M. and Sulaiman, M.N., 2015, A Review on Evaluation Metrics for Data Classification Evaluations International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.5, No.2, March.

Jain, A.P and Katkar, V.D., 2015, Sentiments Analysis Of Twitter Data Using Data Mining , International Conference on Information Processing (ICIP) Vishwakarma Institute of Technology. Dec 16-19

Jagtap, V. S. and Pawar, K., 2013, Sentence-Level Analysis of Sentiment Classification National Conference on Emerging Trends in Engineering, Technology & Architecture

Joshi, R. and Tekchandani, R., 2016, Comparative analysis of Twitter data using supervised classifiers, International Conference on Inventive Computation Technologies (ICICT), Coimbatore, pp. 1-6. doi: 10.1109/INVENTIVE.2016.7830089

KAYNAKLAR DİZİNİ (devam)

Jurafsky, D. and Martin, J.H., 2015, Speech and Language Processing, An Introduction to Natural Language Processing,Computational Linguistics, and Speech Recognition, 65p

Kiruthika, M., Woonna, S. and Giri, P., 2016, Sentiment Analysis of Twitter Data. International Journal of Innovations in Engineering and Technology(IJIET). Volume 6 Issue 4, April, pp. 264 -273, ISSN:2319-1058

Khairnar, J and Kinikar,M., 2013, Machine Learning Algorithms for Opinion Mining and Sentiment Classification International Journal of Scientific and Research Publications, Volume 3, Issue 6

Kolkur, S., Dantal, G. and Mahe, R., 2015, Study of Different Levels for Sentiment Analysis International Journal of Current Engineering and Technology E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Kumar, A and Sebastian,T. M., 2012 ,Sentiment Analysis on Twitter IJCSI International Journal of Computer Science Issues, Vol.9, No 3 July 2012 ISSN (Online): 1694-0814

Liu, B., 2010, Sentiment analysis and subjectweety. Handbook of natural language processing, 2:568

Liu, B., 2012, Sentiment Analysis and Opinion Mining, Morgan & Claypool Publishers, May

Liu, B and Zhang, L., 2012, "A Survey of Opinion Mining and Sentiment Analysis". Sentiment Analysis and Opinion Mining ,Springer, Boston, MA 1-49pp.

Medhat, W., Hassan, A. and Korashy, H., 2014, Sentiment analysis algorithms and applications:A survey Ain Shams Engineering Journal (2014) 5, 1093–1113

Madhoushi, Z., Hamdan, A.R and Zainudin, S., 2015, Sentiment Analysis Techniques in Recent Works Science and Information Conference 2015 July 28-30, London, UK

KAYNAKLAR DİZİNİ (devam)

Patil, P.H., and Atique, M., 2015, Sentiment Analysis for Social Media: A Survey. 2nd International Conference on Information Science and Security (ICISS), Seoul, South Korea

Parrott, W. G., 2001, "Emotions in social psychology: Volume overview." Emotions in social psychology: Essential readings. Ed. W. G. Parrott. Philadelphia: Psychology Press, 2001: pp -38-39

Pak, A and Paroubek, P., 2010, Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of the Seventh Conference on International Language Resources and Evaluation, pp.1320–1326.

Ravi, K. and Ravi, V., 2015, A survey on opinion mining and sentiment analysis:tasks, approaches and applications, Knowledge-Based Systems, vol. 89,pp. 14-46.

Rosen, A. , 2017 "Tweeting Made Easier", https://blog.twitter.com/en_us/topics/product/2017/tweetingmadeeasier.html (Erişim tarihi: 28 Ocak 2019)

Saif, H., 2015, Semantic Sentiment Analysis of Microblogs

Shirsat, V.S., Jagdale, S. S and Deshmukh, S.N, 2017, Document Level Sentiment Analysis from News Articles 978-1-5386-4008-1/17/$31.00 IEEE

Shital, A. P and Jeevan, A.P., 2017, Twitter sentiment classification using stanford NLP. 1st International Conference on Intelligent Systems and Information Management (ICISIM)

Smith, P., 2011, Sentiment Analysis: Beyond Polarity

Velioğlu, R., Yıldız, T. and Yıldırım, S., 2018, Sentiment Analysis Using Learning Approaches Over Emojis for Turkish Tweets. 3rd International Conference on Computer Science and Engineering (UBMK)

Vidushi and Sodhi, G.S., 2017, Sentiment Mining of Online Reviews Using Machine Learning Algorithms IJEDR | Volume 5, Issue 2 | ISSN: 2321-9939

KAYNAKLAR DİZİNİ (devam)

Twitter Statistics, 2019**,** https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/ (Erişim tarihi: 28 Ocak 2019)

Taspinar, A., 2015, Text Classification and Sentiment Analysis http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/ (Erişim tarihi: 12 Şubat 2019)

TEŞEKKÜR

Bu doktora sürecimin her aşamasında büyük desteğini gördüğüm değerli hocam Doç. Dr. Burak ORDİN'e çok teşekkür ederim. Ayrıca, en zorlu zamanlarda desteğini ve ilgisini hiç bir zaman esirgemeyen değerli hocam Prof. Dr. Urfat NURİYEV'e ve Dr. Öğr. Üyesi Samir RUSTAMOV'a teşekkür etmeyi bir borç bilirim.

Beni hayata disiplinli ve sıkı çalışmaya teşvik eden merhum annem Yagut hanıma, akademisyen olarak yetişmem için motive eden babam Nesib beye, eğitim hayatımızda bizlere öncülük eden abim Vurgun beye ve çalışmalarımda desteğini ve hoşgörüsü esirgemeyen sevgili eşim Lale hanıma ve ailemin diğer bireylerine çok teşekkür ederim.

Ayrıca yurt dışında doktora yapabilmem için bana fırsat sunan Azerbaycan Cumhuriyeti Milli Eğitim Bakanlığı'na da teşekkür ederim.

Son olarak doktora tezimi biricik oğlum Nesib`e armağan ediyorum.

ÖZGEÇMİŞ

07.10.1989 tarihinde Azerbaycan'ın Yardımlı ilinin Abidinli köyünde doğdum. İlköğretim ve Lise hayatımı tamamladıktan sonra 2007 yılında, Türkiye Cumhuriyet Sınavı (TCS) sonucunda burslu olarak T. C. Ege Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nü kazandım ve 2012 yılında bu bölümden mezun oldum. Aynı yıl T. C. Ege Üniversitesi Matematik Bölümü Bilgisayar Bilimleri bölümünde yüksek lisans yapmaya başladım ve 2014 yılında bu bölümden mezun oldum. 2014 yılında Ege Üniversitesi Matematik Bölümü Bilgisayar Bilimleri bölümünde doktora eğitimine başladım ve halen doktora eğitimim devam etmektedir. İzmir'de yazılım alanında birçok firmada çalıştım. 2018 yılının Aralık ayından Azerbaycan Cumhuriyeti Cumhurbaşkanlığında yazılım mühendisi ve veri bilimcisi olarak çalışmaktayım.

Makaleler

Hasanli, H., Ordin, B. and Rustamov S., (2019), Sentiment analysis on Twitter data for Azerbaijani language. Journal of Modern Technology and Engineering Vol.4, No.2, 2019, pp.109-121

Uluslararası Konferans Yayınları

Hasanli, H. and Rustamov S., (2019), Sentiment Analysis of Azerbaijani twits Using Logistic Regression, Naive Bayes and SVM. 13th International Conference on Application of Information and Communication Technologies

EKLER

EK 1. Stop Words (Etkisiz Kelimeler)

EK 2. Program kodu

EK 1. Stop Words (Etkisiz Kelimeler)

Azerbaycan	**English**
az	little
arasında	between
bir	a
biri	one of
bu	this
bütün	all
ci	th
cu	th
cü	th
çox	many
da	also
də	also
de	also

Azerbaycan	English
az	little
arasında	between
daha	more
edən	did
edir	is
etmək	do
etmişdir	has
ə	the
ən	most
haqqında	about
hər	each
her	each
xan	khan
görə	according to

Azerbaycan	English
az	little
arasında	between
ki	that
kimi	as
qədər	to
isə	while
il	year
ilk	first
idi	it was
istifadə	use
olan	being
onun	his
olaraq	as
o	it

Azerbaycan	English
az	little
arasında	between
olur	is
onu	it
olmuşdur	was
ona	her
öz	own
sonra	after
tərəfindən	by
üçün	for
var	there
və	and
ve	and
yaxşı	good

Azerbaycan	English
az	little
arasında	between
zaman	time

EK 2. Program kodu

```
namespace PhdTweetApp.ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Program p = new Program();
            p.writeTweetsToDb();
        }
        public void writeTweetsToDb()
        {
            cstmAllTweets c = new cstmAllTweets();
            BussinesLogicOperation blogic = new
BussinesLogicOperation();
            phdSearch psearch = new phdSearch();
            //40.578400, 49.633601
            psearch.latitude = 40.474644;
            psearch.longitude = 49.913266;
            psearch.radius = 300;
            psearch.maximumNumberOfResults = 100000;
            //psearch.maximumNumberOfResults = 100;
            psearch.lang = "Turkish";
            psearch.since = DateTime.Now.AddHours(-24);
            psearch.until = DateTime.Now;
            blogic.prepareTweetsForDB(psearch);
            //
        }
    }
}

namespace PhdTweetApp.BLL
{
    public class BussinesLogicOperation
    {
        static List<tblPozitive> pozitiveWordList = new
List<tblPozitive>();
        static List<tblNegative> negativeWordList = new
List<tblNegative>();
```

```
            static List<tblEmoji> emojiWordList = new
List<tblEmoji>();
            BussinesLogicRepository bRepository = new
BussinesLogicRepository();
            public BussinesLogicOperation()
            {
                var user = Login.LoginUser();
                emojiWordList = bRepository.gettblEmojis();
                negativeWordList = bRepository.gettblNegatives();
                pozitiveWordList = bRepository.gettblPozitives();
            }

            public List<cstmAllTweets>
getTweetListForDB(IEnumerable<ITweet> tweets)
            {
                List<cstmAllTweets> list = new
List<cstmAllTweets>();
                if (tweets != null)
                {
                    foreach (var item in tweets)
                    {
                        if
(TweetUtil.ControlTweet(item.CreatedBy.Location, item.Retweeted))
                        {
                            cstmAllTweets obj = new
cstmAllTweets();
                            tblAllTweet pTweet = new
tblAllTweet();
                            List<tblAllTweetHashtag> phashtagList
= new List<tblAllTweetHashtag>();
                            if (item.Hashtags.Count > 0)
                            {
                                foreach (var item2 in
item.Hashtags)
                                {
                                    pTweet.Hashtags =
pTweet.Hashtags + item2.Text + " ; ";

                                    tblAllTweetHashtag pHashtag =
new tblAllTweetHashtag();
```

```
                                        pHashtag.Hashtag =
item2.Text;
                                        phashtagList.Add(pHashtag);
                                    }
                                }
                                pTweet.Language =
item.Language.ToString();
                                pTweet.TweetLocalCreationDate =
item.TweetLocalCreationDate;
                                pTweet.CreatedAt = item.CreatedAt;

                                pTweet.UserName =
item.CreatedBy.Name;
                                pTweet.UserDTO =
item.CreatedBy.UserDTO.ToString();
                                pTweet.UserLocation =
item.CreatedBy.Location;
                                pTweet.FullText = item.FullText;

                                string _cleanText =
TweetUtil.removeHastagAndOtherTags(item.FullText);
                                pTweet.CleanText = _cleanText;

                                string _updateText =
EmojiConverter.updateSentence(_cleanText);
                                string _convertedText =
EmojiConverter.UnicodeSplitData(_updateText);
                                pTweet.ConvertedText =
_convertedText;
                                obj.allTweet = pTweet;
                                obj.allTweetHashtag = phashtagList;
                                list.Add(obj);
                            }
                        }

                    }

                    return list;
                }
```

```
            public void prepareTweetsForDB(phdSearch psearch)
            {
                List<cstmAllTweets> list = new
List<cstmAllTweets>();

                var tweets = psearch.getAllTweets(psearch);

                //foreach (var item in tweets)
                //{
                //    Console.WriteLine("Name
"+item.CreatedBy.UserDTO.Name +"  ScreenName: " +
item.CreatedBy.ScreenName + "  Location: " +
item.CreatedBy.Location);
                //}
                phdTweet ptweet = new phdTweet();

                list = getTweetListForDB(tweets);



                BussinesLogicRepository blogicRepo = new
BussinesLogicRepository();
                foreach (var item in list)
                {

                    blogicRepo.addCstmAllTweets(item);



                }

            }


            public void
getDailyTweetHastag(List<tblAllTweetHashtag> list)
            {

                foreach (var item in list)
                {
```

```
                    searchByHashTag(item.Hashtag);
                }
            }

            public void searchByHashTag(string hashtag)
            {

                ICoordinates coor = new Coordinates(40.409264,
49.867092);
                IGeoCode geoCode = new GeoCode(coor, 10,
DistanceMeasure.Kilometers);
                var searchParameter = new
SearchTweetsParameters(hashtag);
                searchParameter.TweetSearchType =
TweetSearchType.OriginalTweetsOnly;
                searchParameter.GeoCode = geoCode;
                searchParameter.Lang = LanguageFilter.Turkish;
                int hour = DateTime.Now.Hour;
                /// searchParameter.Since =
DateTime.Now.AddHours(-hour);
                // searchParameter.Until =
DateTime.Now.AddDays(1);
                var testtweets =
Search.SearchTweets(searchParameter);

                if (testtweets != null)
                {

                    string path =
@"D:\Doktora\Tweetinvi\TweetinviClient\tweets.txt";
                    foreach (var item1 in testtweets)
                    {

                        if
(item1.CreatedBy.Location.Contains("zərbaycan") ||
item1.CreatedBy.Location.Contains("aku") ||
item1.CreatedBy.Location.Contains("aку") ||
item1.CreatedBy.Location.Contains("aki") ||
item1.CreatedBy.Location.Contains("akı") ||
item1.CreatedBy.Location.Contains("zerbaijan") ||
item1.CreatedBy.Location.Contains("зербайджан"))
```

```
                {


                }
            }

        }

    }

    public static int getWordScore(string word)
    {
        int score = 0;
        if (!string.IsNullOrEmpty(word))
        {
            string[] array = word.Split(' ');
            for (int i = 0; i < array.Length; i++)
            {
                string item = array[i];
                if (!string.IsNullOrWhiteSpace(item))
                {

                    bool flag = true;

                    tblPozitive p1 =
pozitiveWordList.Where(x => x.Word.ToLower() ==
item).FirstOrDefault();
                    if (p1 != null)
                    {
                        score = score + (int)p1.Value;
                        flag = false;
                    }

                    if (flag)
                    {
```

```
                                    tblNegative n1 =
negativeWordList.Where(x => x.Word.ToLower() ==
item).FirstOrDefault();
                                    if (n1 != null)
                                    {
                                        score = score -
(int)n1.Value;
                                        flag = false;
                                    }
                                }
                                if (flag)
                                {
                                    tblEmoji e1 =
emojiWordList.Where(x => x.Sequence.ToLower() ==
item).FirstOrDefault();
                                    if (e1 != null)
                                    {
                                        score = score +
(int)e1.Value;
                                        flag = false;
                                    }
                                }
                            }


                        }

                    }

                    return score;


                }

            }
        }
        namespace PhdTweetApp.BLL
        {
            public class BussinesLogicRepository
            {
                OperationLogic ologic = new OperationLogic();
```

```
        public bool addCstmAllTweets(cstmAllTweets item)
        {

            try
            {
                ologic.addCstmAllTweets(item);
                return true;
            }
            catch (Exception ex)
            {
                return false;
            }



        }
        public List<tblRegion> getRegions()
        {

            try
            {
                return ologic.gettblRegion();
            }
            catch (Exception ex)
            {
                return null;
            }


        }
        public List<tblRegion> getRegionsByParentCode(string
parentCode)
        {
            try
            {
                return
ologic.gettblRegionsByParentCode(parentCode);
            }
            catch (Exception ex)
```

```
        {
            return null;
        }

    }

    public tblRegion getRegionByCode(string Code)
    {
        try
        {
            return ologic.gettblRegionByCode(Code);
        }
        catch (Exception ex)
        {
            return null;
        }
    }

    public List<tblNegative> gettblNegatives()
    {

        try
        {
            return ologic.gettblNegatives();
        }
        catch (Exception ex)
        {
            return null;
        }

    }

    public List<tblPozitive> gettblPozitives()
    {

        try
        {
            return ologic.gettblPozitives();
        }
        catch (Exception ex)
        {
```

```
                return null;
            }

        }

        public List<tblEmoji> gettblEmojis()
        {

            try
            {
                return ologic.gettblEmojis();
            }
            catch (Exception ex)
            {
                return null;
            }

        }

    }

}

namespace PhdTweetApp.BLL.Model
{
    public class phdTweet
    {

        public string FullText { get; set; }
        public string CleanText { get; set; }
        public string ConvertedText { get; set; }
        public string Cost { get; set; }
        public string Sentiment { get; set; }
        public string MainLanguage { get; set; }
        public string UserDTO { get; set; }
        public string UserName { get; set; }
        public string UserLocation { get; set; }
        public string Language { get; set; }
        public string Hashtags { get; set; }
        public DateTime TweetLocalCreationDate { get; set; }
        public DateTime CreatedAt { get; set; }
```

```
            public int LexiconSentScore { get; set; }



            public List<phdTweet>
getTweetList(IEnumerable<ITweet> tweets)
            {
                List<phdTweet> list = new List<phdTweet>();


                if (tweets != null)
                {
                    foreach (var item in tweets)
                    {
                        if
(TweetUtil.ControlTweet(item.CreatedBy.Location, item.IsRetweet))
                        {

                            phdTweet pTweet = new phdTweet();


                            if (item.Hashtags.Count > 0)
                            {
                                foreach (var item2 in
item.Hashtags)
                                {
                                    pTweet.Hashtags =
pTweet.Hashtags + item2.Text + " ; ";
                                }

                            }
                            pTweet.Language =
item.Language.ToString();
                            pTweet.TweetLocalCreationDate =
item.TweetLocalCreationDate;
                            pTweet.CreatedAt = item.CreatedAt;

                            pTweet.UserName =
item.CreatedBy.Name;
```

```
                            pTweet.UserDTO =
item.CreatedBy.UserDTO.ToString();
                            pTweet.UserLocation =
item.CreatedBy.Location;
                            pTweet.FullText = item.FullText;

                            string _cleanText =
TweetUtil.removeHastagAndOtherTags(item.FullText);
                            pTweet.CleanText = _cleanText;

                            string _updateText =
EmojiConverter.updateSentence(_cleanText);
                            string _convertedText=
EmojiConverter.UnicodeSplitData(_updateText);
                            pTweet.ConvertedText =
_convertedText;
                            pTweet.LexiconSentScore =
TweetUtil.GetWordScore(_convertedText);
                            //pTweet.ConvertedText =
EmojiConverter.UnicodeSplitData(local.CleanText);
                            list.Add(pTweet);

                        }
                    }
                }

                return list;
            }



        }
    }

    namespace PhdTweetApp.BLL.Model
    {
       public class phdSearch
        {
            public double latitude { get; set; }
            public double longitude { get; set; }
            public double radius { get; set; }
```

```
            public string lang { get; set; }
            public int maximumNumberOfResults { get; set; }
            public DateTime since { get; set; }
            public DateTime until { get; set; }


            public phdSearch() { }
            public phdSearch(double _latitude, double _longitude,
double _radius, string _lang, int _mNumber, DateTime _since,
DateTime _until)
            {
                this.latitude = _latitude;
                this.longitude = _longitude;
                this.radius = _radius;
                this.lang = _lang;
                this.maximumNumberOfResults = _mNumber;
                this.since = _since;
                this.until = _until;
            }

            public IEnumerable<ITweet> getAllTweets(phdSearch
srch)
            {

                if (srch != null)
                {
                    // Complex search
                    var searchParameter =
Search.CreateTweetSearchParameter("");

                    if (srch.latitude != 0 && srch.longitude != 0
&& srch.radius != 0)
                    {
                        searchParameter.SetGeoCode(srch.latitude,
srch.longitude, srch.radius, DistanceMeasure.Kilometers);
                    }
                    if (!String.IsNullOrEmpty(srch.lang))
                    {
                        if (srch.lang.Equals("English"))
                        {
```

```
                            searchParameter.Lang =
LanguageFilter.English;
                        }
                        else if (srch.lang.Equals("Turkish"))
                        {
                            searchParameter.Lang =
LanguageFilter.Turkish;
                        }
                        else if (srch.lang.Equals("Russian"))
                        {
                            searchParameter.Lang =
LanguageFilter.Russian;
                        }

                    }
                    if (srch.maximumNumberOfResults != 0)
                    {

                        searchParameter.MaximumNumberOfResults =
srch.maximumNumberOfResults;
                    }


                    //searchParameter.Since = new DateTime(2017,
5, 1);
                  //  searchParameter.Until = new DateTime(2017,
7, 1);
                    searchParameter.Until = srch.until;
                    searchParameter.Since = srch.since;
                    //  searchParameter.SinceId =
399616835892781056;
                    //  searchParameter.MaxId =
405001488843284480;
                    var testtweets =
Search.SearchTweets(searchParameter);



                    return testtweets;
                }
                else
```

```
                {
                    return null;
                }

            }


        }
    }


    namespace PhdTweetApp.BLL.Helper
    {
        public class EmojiConverter
        {
            public static string UnicodeSplitData(string
wordData)
            {
                //Dictionary<int, string> dictEmojiIndex = new
Dictionary<int, string>();
                List<EmojiObj> emojiIndexList = new
List<EmojiObj>();
                wordData =
EmojiConverter.updateSentence(wordData);
                // string testdata = "tebrik😀";
                string temp = string.Empty;
                for (int i = 0; i < wordData.Length; i++)
                {
                    temp = temp + wordData[i].ToString();
                    try
                    {

                        var codepoints = temp.Codepoints();
                        var sequence = new
UnicodeSequence(codepoints);
                        temp = string.Empty;

                        bool flag =
Emoji.IsEmoji(sequence.AsString);
                        if (flag)
                        {
```

```
                            //Console.WriteLine("Gridi");

//Console.WriteLine(sequence.AsString);



                            var data =
sequence.Codepoints.ToList();
                            var dataObj = data[0].ToString();
                            string code = dataObj.Substring(2);


                            #region subgroup: face-positive


                            // 1F600; fully - qualified     # 😀
grinning face
                            if (sequence.AsString ==
Emoji.GrinningFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.GrinningFace.Name;
                                emojiIndexList.Add(eobj);
                            }
                            // 1F601 ; fully-qualified     # 😁
grinning face with smiling eyes
                            else if (sequence.AsString ==
Emoji.GrinningFaceWithSmilingEyes.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.GrinningFaceWithSmilingEyes.Name;
                                emojiIndexList.Add(eobj);
                            }
```

```
                                //1F602; fully - qualified     # 😂
face with tears of joy
                                else if (sequence.AsString ==
Emoji.FaceWithTearsOfJoy.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceWithTearsOfJoy.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F923; fully - qualified     # 🤣
rolling on the floor laughing
                                else if (sequence.AsString ==
Emoji.RollingOnTheFloorLaughing.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.RollingOnTheFloorLaughing.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F604; fully - qualified     # 😄
smiling face with open mouth & smiling eyes
                                else if (sequence.AsString ==
Emoji.SmilingFaceWithOpenMouthAndSmilingEyes.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFaceWithOpenMouthAndSmilingEyes.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                // 1F605; fully - qualified     # 😅
smiling face with open mouth & cold sweat
                                else if (sequence.AsString ==
Emoji.SmilingFaceWithOpenMouthAndColdSweat.ToString())
```

```
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFaceWithOpenMouthAndColdSweat.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F606; fully - qualified     # 😆
smiling face with open mouth & closed eyes
                                else if (sequence.AsString ==
Emoji.SmilingFaceWithOpenMouthAndClosedEyes.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFaceWithOpenMouthAndClosedEyes.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F609; fully - qualified     # 😉
winking face
                                else if (sequence.AsString ==
Emoji.WinkingFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.WinkingFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F60A; fully - qualified     # 😊
smiling face with smiling eyes
                                else if (sequence.AsString ==
Emoji.SmilingFaceWithSmilingEyes.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
```

```
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFaceWithSmilingEyes.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //1F60B; fully-qualified     # 😋
face savouring delicious food
                                else if (sequence.AsString ==
Emoji.FaceSavouringDeliciousFood.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceSavouringDeliciousFood.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F60E ; fully-qualified     # 😎
smiling face with sunglasses
                                else if (sequence.AsString ==
Emoji.SmilingFaceWithSunglasses.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFaceWithSunglasses.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F60D; fully - qualified     # 😍
smiling face with heart-eyes
                                else if (sequence.AsString ==
Emoji.SmilingFaceWithHeartEyes.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFaceWithHeartEyes.Name;
```

```
                                emojiIndexList.Add(eobj);
                            }
                            //1F618; fully - qualified     # 😘
face blowing a kiss
                            else if (sequence.AsString ==
Emoji.FaceBlowingAKiss.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceBlowingAKiss.Name;
                                emojiIndexList.Add(eobj);
                            }
                            //1F617; fully - qualified     # 😗
kissing face
                            else if (sequence.AsString ==
Emoji.KissingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.KissingFace.Name;
                                emojiIndexList.Add(eobj);
                            }
                            // 1F619; fully - qualified     # 😙
kissing face with smiling eyes
                            else if (sequence.AsString ==
Emoji.KissingFaceWithSmilingEyes.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.KissingFaceWithSmilingEyes.Name;
                                emojiIndexList.Add(eobj);
                            }
                            // 1F61A; fully - qualified     # 😚
kissing face with closed eyes
```

```
                                else if (sequence.AsString ==
Emoji.KissingFaceWithClosedEyes.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.KissingFaceWithClosedEyes.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                // 263A FE0F; fully - qualified     #
☺️ smiling face
                                else if (sequence.AsString ==
Emoji.SmilingFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFace.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                // 263A; non - fully - qualified # ☺
smiling face
                                else if (sequence.AsString ==
Emoji.SmilingFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SmilingFace.Name;
                                    emojiIndexList.Add(eobj);
                                }
                                //1F642; fully - qualified     # ☺
slightly smiling face
                                else if (sequence.AsString ==
Emoji.SlightlySmilingFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
```

```
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SlightlySmilingFace.Name;
                                emojiIndexList.Add(eobj);
                            }
                            //1F917 ; fully-qualified     # 🤗
hugging face
                            else if (sequence.AsString ==
Emoji.HuggingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.HuggingFace.Name;
                                emojiIndexList.Add(eobj);
                            }
                            //1F929; fully - qualified     # 🤩
star-struck
                            else if (sequence.AsString ==
Emoji.StarStruck.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.StarStruck.Name;
                                emojiIndexList.Add(eobj);
                            }

                            #endregion

                            #region subgroup: face-neutral


                            // 1F914; fully - qualified     # 🤔
thinking face
                            else if (sequence.AsString ==
Emoji.ThinkingFace.ToString())
```

```
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.ThinkingFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F928; fully - qualified     # 🤨
face with raised eyebrow
                                else if (sequence.AsString ==
Emoji.FaceWithRaisedEyebrow.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceWithRaisedEyebrow.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F610; fully - qualified     # 😐
neutral face
                                else if (sequence.AsString ==
Emoji.NeutralFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.NeutralFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F611; fully - qualified     # 😑
expressionless face
                                else if (sequence.AsString ==
Emoji.ExpressionlessFace.ToString())
                                {
```

```
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ExpressionlessFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F636; fully - qualified     # 😶
face without mouth
                            else if (sequence.AsString ==
Emoji.FaceWithoutMouth.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithoutMouth.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F644; fully - qualified     # 🙄
face with rolling eyes
                            else if (sequence.AsString ==
Emoji.FaceWithRollingEyes.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithRollingEyes.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F623; fully - qualified     # 😣
persevering face
                            else if (sequence.AsString ==
Emoji.PerseveringFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
```

```
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.PerseveringFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F625; fully - qualified     # 😥
disappointed but relieved face
                            else if (sequence.AsString ==
Emoji.DisappointedButRelievedFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.DisappointedButRelievedFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F62E; fully - qualified     # 😮
face with open mouth
                            else if (sequence.AsString ==
Emoji.FaceWithOpenMouth.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithOpenMouth.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F910; fully - qualified     # 🤐
zipper-mouth face
                            else if (sequence.AsString ==
Emoji.ZipperMouthFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
```

```
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ZipperMouthFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F62F; fully - qualified     # 😯
hushed face
                            else if (sequence.AsString ==
Emoji.HushedFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.HushedFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F62A; fully - qualified     # 😪
sleepy face
                            else if (sequence.AsString ==
Emoji.SleepyFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SleepyFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F62B; fully - qualified     # 😫
tired face
                            else if (sequence.AsString ==
Emoji.TiredFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
```

```
                                    eobj.EmojiName =
Emoji.TiredFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F634; fully - qualified     # 😴
sleeping face
                                else if (sequence.AsString ==
Emoji.SleepingFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SleepingFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F60C; fully - qualified     # 😌
relieved face
                                else if (sequence.AsString ==
Emoji.RelievedFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.RelievedFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F61B; fully - qualified     # 😛
face with stuck-out tongue
                                else if (sequence.AsString ==
Emoji.FaceWithStuckOutTongue.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
```

```
                                eobj.EmojiName =
Emoji.FaceWithStuckOutTongue.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F61C; fully - qualified     # 😜
face with stuck-out tongue & winking eye
                            else if (sequence.AsString ==
Emoji.FaceWithStuckOutTongueAndWinkingEye.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithStuckOutTongueAndWinkingEye.Name;
                                emojiIndexList.Add(eobj);

                            }


                            // 1F61D; fully - qualified     # 😝
face with stuck-out tongue & closed eyes
                            else if (sequence.AsString ==
Emoji.FaceWithStuckOutTongueAndClosedEyes.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithStuckOutTongueAndClosedEyes.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F924; fully - qualified     # 🤤
drooling face
                            else if (sequence.AsString ==
Emoji.DroolingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
```

```
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.DroolingFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F612; fully - qualified     # 😒
unamused face
                                else if (sequence.AsString ==
Emoji.UnamusedFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.UnamusedFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //  1F613; fully - qualified     # 😓
face with cold sweat
                                else if (sequence.AsString ==
Emoji.FaceWithColdSweat.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceWithColdSweat.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //  1F614; fully - qualified     # 😔
pensive face
                                else if (sequence.AsString ==
Emoji.PensiveFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
```

```
                                eobj.EmojiName =
Emoji.PensiveFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F615; fully - qualified     # 😕
confused face
                            else if (sequence.AsString ==
Emoji.ConfusedFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ConfusedFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F643; fully - qualified     # 🙃
upside-down face
                            else if (sequence.AsString ==
Emoji.UpsideDownFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.UpsideDownFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F911; fully - qualified     # 🤑
money-mouth face
                            else if (sequence.AsString ==
Emoji.MoneyMouthFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
```

```
                                    eobj.EmojiName =
Emoji.MoneyMouthFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //1F632; fully - qualified     # 😲
astonished face
                                else if (sequence.AsString ==
Emoji.AstonishedFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.AstonishedFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                #endregion

                                #region subgroup: face-negative


                                //2639 FE0F; fully - qualified     #
☹ frowning face
                                else if (sequence.AsString ==
Emoji.FrowningFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FrowningFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 2639; non - fully - qualified # ☹
frowning face
                                else if (sequence.AsString ==
Emoji.FrowningFace.ToString())
```

```
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FrowningFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F641; fully - qualified     # ☹
slightly frowning face
                                else if (sequence.AsString ==
Emoji.SlightlyFrowningFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SlightlyFrowningFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F616; fully - qualified     # 😖
confounded face
                                else if (sequence.AsString ==
Emoji.ConfoundedFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.ConfoundedFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F61E; fully - qualified     # 😞
disappointed face
                                else if (sequence.AsString ==
Emoji.DisappointedFace.ToString())
                                {
```

```
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.DisappointedFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F61F; fully - qualified     # 😟
worried face
                            else if (sequence.AsString ==
Emoji.WorriedFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.WorriedFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F624; fully - qualified     # 😤
face with steam from nose
                            else if (sequence.AsString ==
Emoji.FaceWithSteamFromNose.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithSteamFromNose.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F622; fully - qualified     # 😢
crying face
                            else if (sequence.AsString ==
Emoji.CryingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
```

```
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.CryingFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F62D; fully - qualified     # 😭
loudly crying face
                            else if (sequence.AsString ==
Emoji.LoudlyCryingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.LoudlyCryingFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F626; fully - qualified     # 😦
frowning face with open mouth
                            else if (sequence.AsString ==
Emoji.FrowningFaceWithOpenMouth.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FrowningFaceWithOpenMouth.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F627; fully - qualified     # 😧
anguished face
                            else if (sequence.AsString ==
Emoji.AnguishedFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
```

```
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.AnguishedFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F628; fully - qualified     # 😨
fearful face
                            else if (sequence.AsString ==
Emoji.FearfulFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FearfulFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F629; fully - qualified     # 😩
weary face
                            else if (sequence.AsString ==
Emoji.WearyFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.WearyFace.Name;
                                emojiIndexList.Add(eobj);

                            }

                            //1F92F; fully - qualified     # 🤯
exploding head
                            else if (sequence.AsString ==
Emoji.ExplodingHead.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
```

```
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ExplodingHead.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F62C; fully - qualified     # 😬
grimacing face
                            else if (sequence.AsString ==
Emoji.GrimacingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.GrimacingFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F630; fully - qualified     # 😰
face with open mouth & cold sweat
                            else if (sequence.AsString ==
Emoji.FaceWithOpenMouthAndColdSweat.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithOpenMouthAndColdSweat.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F631; fully - qualified     # 😱
face screaming in fear
                            else if (sequence.AsString ==
Emoji.FaceScreamingInFear.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
```

```
                                    eobj.EmojiName =
Emoji.FaceScreamingInFear.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //1F633; fully - qualified     # 😳
flushed face
                                else if (sequence.AsString ==
Emoji.FlushedFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FlushedFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //1F92A; fully - qualified     # 🤪
crazy face
                                else if (sequence.AsString ==
Emoji.CrazyFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.CrazyFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //1F635; fully - qualified     # 😵
dizzy face
                                else if (sequence.AsString ==
Emoji.DizzyFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
```

```
                                eobj.EmojiName =
Emoji.DizzyFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F621; fully - qualified     # 😡
pouting face
                            else if (sequence.AsString ==
Emoji.PoutingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.PoutingFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F620; fully - qualified     # 😠
angry face
                            else if (sequence.AsString ==
Emoji.AngryFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.AngryFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F92C; fully - qualified     # 🤬
face with symbols over mouth
                            else if (sequence.AsString ==
Emoji.FaceWithSymbolsOverMouth.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
```

```
                                    eobj.EmojiName =
Emoji.FaceWithSymbolsOverMouth.Name;
                                    emojiIndexList.Add(eobj);

                                }

                                #endregion

                                #region subgroup: face-sick

                                //  1F637; fully - qualified     # 😷
face with medical mask
                                else if (sequence.AsString ==
Emoji.FaceWithMedicalMask.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceWithMedicalMask.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //  1F912; fully - qualified     # 🤒
face with thermometer
                                else if (sequence.AsString ==
Emoji.FaceWithThermometer.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceWithThermometer.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F915; fully - qualified     # 🤕
face with head-bandage
                                else if (sequence.AsString ==
Emoji.FaceWithHeadBandage.ToString())
```

```
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceWithHeadBandage.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F922; fully - qualified     # 🤢
nauseated face
                                else if (sequence.AsString ==
Emoji.NauseatedFace.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.NauseatedFace.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F92E; fully - qualified     # 🤮
face vomiting
                                else if (sequence.AsString ==
Emoji.FaceVomiting.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FaceVomiting.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F927; fully - qualified     # 🤧
sneezing face
                                else if (sequence.AsString ==
Emoji.SneezingFace.ToString())
                                {
```

```
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SneezingFace.Name;
                                emojiIndexList.Add(eobj);

                            }

                            #endregion

                            #region subgroup: face-role


                            //   1F607; fully - qualified     #
😇 smiling face with halo
                            else if (sequence.AsString ==
Emoji.SmilingFaceWithHalo.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SmilingFaceWithHalo.Name;
                                emojiIndexList.Add(eobj);

                            }

                            //  1F920; fully - qualified     # 🤠
cowboy hat face
                            else if (sequence.AsString ==
Emoji.CowboyHatFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.CowboyHatFace.Name;
                                emojiIndexList.Add(eobj);

                            }
```

```
                            //  1F921; fully - qualified     # 🤡
clown face
                            else if (sequence.AsString ==
Emoji.ClownFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ClownFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F925; fully - qualified     #
🤥 lying face
                            else if (sequence.AsString ==
Emoji.LyingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.LyingFace.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F92B; fully - qualified     # 🤫
shushing face
                            else if (sequence.AsString ==
Emoji.ShushingFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ShushingFace.Name;
                                emojiIndexList.Add(eobj);

                            }
```

```
                            //   1F92D; fully - qualified     #
🤭 face with hand over mouth
                            else if (sequence.AsString ==
Emoji.FaceWithHandOverMouth.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithHandOverMouth.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F9D0; fully-qualified     # 🧐
face with monocle
                            else if (sequence.AsString ==
Emoji.FaceWithMonocle.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.FaceWithMonocle.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //    1F913; fully - qualified     #
🤓 nerd face
                            else if (sequence.AsString ==
Emoji.NerdFace.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.NerdFace.Name;
                                emojiIndexList.Add(eobj);

                            }
```

```
                            #endregion

                            #region subgroup: plant-flower

                            // 1F490; fully - qualified     # 💐
bouquet
                            else if (sequence.AsString ==
Emoji.Bouquet.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Bouquet.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F338; fully - qualified     # 🌸
cherry blossom
                            else if (sequence.AsString ==
Emoji.CherryBlossom.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.CherryBlossom.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F4AE; fully - qualified     # 💮
white flower
                            else if (sequence.AsString ==
Emoji.WhiteFlower.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.WhiteFlower.Name;
```

```
                emojiIndexList.Add(eobj);

            }
            //   1F3F5 FE0F; fully - qualified
# 🏵 rosette
            else if (sequence.AsString ==
Emoji.Rosette.ToString())
            {
                EmojiObj eobj = new EmojiObj();
                eobj.Index = i;
                eobj.EmojiCode = code;
                eobj.EmojiName =
Emoji.Rosette.Name;
                emojiIndexList.Add(eobj);

            }
            //   1F3F5; non-fully-qualified # 🏵
rosette
            else if (sequence.AsString ==
Emoji.Rosette.ToString())
            {
                EmojiObj eobj = new EmojiObj();
                eobj.Index = i;
                eobj.EmojiCode = code;
                eobj.EmojiName =
Emoji.Rosette.Name;
                emojiIndexList.Add(eobj);

            }
            // 1F339  ; fully-qualified     # 🌹
rose
            else if (sequence.AsString ==
Emoji.Rose.ToString())
            {
                EmojiObj eobj = new EmojiObj();
                eobj.Index = i;
                eobj.EmojiCode = code;
                eobj.EmojiName = Emoji.Rose.Name;
                emojiIndexList.Add(eobj);
```

```
                            }

                            //1F940; fully - qualified     # 🥀
wilted flower
                            else if (sequence.AsString ==
Emoji.WiltedFlower.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.WiltedFlower.Name;
                                emojiIndexList.Add(eobj);

                            }

                            //1F33A; fully-qualified     # 🌺
hibiscus
                            else if (sequence.AsString ==
Emoji.Hibiscus.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Hibiscus.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F33B; fully-qualified     # 🌻
sunflower
                            else if (sequence.AsString ==
Emoji.Sunflower.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Sunflower.Name;
                                emojiIndexList.Add(eobj);
```

```
                        }

                        //1F33C; fully-qualified     # ✿
blossom
                        else if (sequence.AsString ==
Emoji.Blossom.ToString())
                        {
                            EmojiObj eobj = new EmojiObj();
                            eobj.Index = i;
                            eobj.EmojiCode = code;
                            eobj.EmojiName =
Emoji.Blossom.Name;
                            emojiIndexList.Add(eobj);

                        }
                        //1F337; fully-qualified     # ✿
tulip
                        else if (sequence.AsString ==
Emoji.Tulip.ToString())
                        {
                            EmojiObj eobj = new EmojiObj();
                            eobj.Index = i;
                            eobj.EmojiCode = code;
                            eobj.EmojiName =
Emoji.Tulip.Name;
                            emojiIndexList.Add(eobj);

                        }



                        #endregion

                        #region subgroup: plant-other

                        //  1F331; fully - qualified     # 🌱
seedling
                        else if (sequence.AsString ==
Emoji.Seedling.ToString())
```

```
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Seedling.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F332; fully - qualified     # 🌲
evergreen tree
                            else if (sequence.AsString ==
Emoji.EvergreenTree.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.EvergreenTree.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //    1F333; fully - qualified     #
🌳 deciduous tree
                            else if (sequence.AsString ==
Emoji.DeciduousTree.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.DeciduousTree.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F334; fully - qualified     #
🌴 palm tree
                            else if (sequence.AsString ==
Emoji.PalmTree.ToString())
                            {
```

```
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.PalmTree.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F335; fully - qualified     # 🌵
cactus
                            else if (sequence.AsString ==
Emoji.Cactus.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Cactus.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //    1F33E; fully - qualified     #
🌾 sheaf of rice
                            else if (sequence.AsString ==
Emoji.SheafOfRice.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SheafOfRice.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F33F; fully - qualified     # 🌿
herb
                            else if (sequence.AsString ==
Emoji.Herb.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
```

```
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName = Emoji.Herb.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //    2618 FE0F; fully - qualified
# ☘ shamrock
                            else if (sequence.AsString ==
Emoji.Shamrock.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Shamrock.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 2618; non - fully - qualified # ☘
shamrock
                            else if (sequence.AsString ==
Emoji.Shamrock.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Shamrock.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F340; fully - qualified     # 🍀
four leaf clover
                            else if (sequence.AsString ==
Emoji.FourLeafClover.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
```

```
                                    eobj.EmojiName =
Emoji.FourLeafClover.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //  1F341; fully - qualified     # 🍁
maple leaf
                                else if (sequence.AsString ==
Emoji.MapleLeaf.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.MapleLeaf.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //   1F342; fully - qualified     #
🍂 fallen leaf
                                else if (sequence.AsString ==
Emoji.FallenLeaf.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.FallenLeaf.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //    1F343; fully - qualified     #
🍃 leaf fluttering in wind
                                else if (sequence.AsString ==
Emoji.LeafFlutteringInWind.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
```

```
                                eobj.EmojiName =
Emoji.LeafFlutteringInWind.Name;
                                emojiIndexList.Add(eobj);

                            }
                            #endregion

                            #region subgroup: emotion


                            //  1F48B; fully - qualified     # 💋
kiss mark
                            else if (sequence.AsString ==
Emoji.KissMark.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.KissMark.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F498; fully - qualified     # 💘
heart with arrow
                            else if (sequence.AsString ==
Emoji.HeartWithArrow.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.HeartWithArrow.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  2764 FE0F; fully - qualified
# ❤️ red heart
                            else if (sequence.AsString ==
Emoji.RedHeart.ToString())
```

```
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.RedHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   2764; non - fully - qualified #
❤ red heart
                            else if (sequence.AsString ==
Emoji.RedHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.RedHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F493; fully - qualified     # 💓
beating heart
                            else if (sequence.AsString ==
Emoji.Seedling.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Seedling.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F494; fully - qualified     #
💔 broken heart
                            else if (sequence.AsString ==
Emoji.BrokenHeart.ToString())
                            {
```

```
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.BrokenHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F495; fully - qualified     #
💕 two hearts
                            else if (sequence.AsString ==
Emoji.TwoHearts.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.TwoHearts.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F496; fully - qualified     # 💖
sparkling heart
                            else if (sequence.AsString ==
Emoji.SparklingHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SparklingHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F497; fully - qualified     # 💗
growing heart
                            else if (sequence.AsString ==
Emoji.GrowingHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
```

```
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.GrowingHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F499; fully - qualified     #
💙 blue heart
                            else if (sequence.AsString ==
Emoji.BlueHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.BlueHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F49A; fully - qualified     #
💚 green heart
                            else if (sequence.AsString ==
Emoji.GreenHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.GreenHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //    1F49B; fully - qualified     #
💛 yellow heart
                            else if (sequence.AsString ==
Emoji.YellowHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
```

```
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.YellowHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F9E1; fully - qualified     # 🧡
orange heart
                            else if (sequence.AsString ==
Emoji.OrangeHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.OrangeHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //       1F49C; fully - qualified
# 💜 purple heart
                            else if (sequence.AsString ==
Emoji.PurpleHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.PurpleHeart.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //   1F5A4; fully - qualified     #
🖤 black heart
                            else if (sequence.AsString ==
Emoji.BlackHeart.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
```

```
                                    eobj.EmojiName =
Emoji.BlackHeart.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //     1F49D; fully - qualified     #
💝 heart with ribbon
                                else if (sequence.AsString ==
Emoji.HeartWithRibbon.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.HeartWithRibbon.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //  1F49E; fully - qualified     # 💞
revolving hearts
                                else if (sequence.AsString ==
Emoji.RevolvingHearts.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.BlueHeart.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F49F; fully - qualified     # 💟
heart decoration
                                else if (sequence.AsString ==
Emoji.HeartDecoration.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
```

```
                                eobj.EmojiName =
Emoji.HeartDecoration.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //2763 FE0F; fully - qualified     #
❣ heavy heart exclamation
                            else if (sequence.AsString ==
Emoji.HeavyHeartExclamation.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.HeavyHeartExclamation.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //    2763; non - fully - qualified #
❣ heavy heart exclamation
                            else if (sequence.AsString ==
Emoji.HeavyHeartExclamation.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.HeavyHeartExclamation.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F48C; fully - qualified     # 💌
love letter
                            else if (sequence.AsString ==
Emoji.LoveLetter.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.LoveLetter.Name;
```

```
                    emojiIndexList.Add(eobj);

                }
                // 1F4A4; fully - qualified     # 💤
zzz
                else if (sequence.AsString ==
Emoji.Zzz.ToString())
                {
                    EmojiObj eobj = new EmojiObj();
                    eobj.Index = i;
                    eobj.EmojiCode = code;
                    eobj.EmojiName = Emoji.Zzz.Name;
                    emojiIndexList.Add(eobj);

                }
                // 1F4A2; fully - qualified     # 💢
anger symbol
                else if (sequence.AsString ==
Emoji.AngerSymbol.ToString())
                {
                    EmojiObj eobj = new EmojiObj();
                    eobj.Index = i;
                    eobj.EmojiCode = code;
                    eobj.EmojiName =
Emoji.AngerSymbol.Name;
                    emojiIndexList.Add(eobj);

                }

                // 1F4A3; fully - qualified     # 💣
bomb
                else if (sequence.AsString ==
Emoji.Bomb.ToString())
                {
                    EmojiObj eobj = new EmojiObj();
                    eobj.Index = i;
                    eobj.EmojiCode = code;
                    eobj.EmojiName = Emoji.Bomb.Name;
                    emojiIndexList.Add(eobj);
```

```
                            }
                            // 1F4A5; fully - qualified     # 💥
collision
                            else if (sequence.AsString ==
Emoji.Collision.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.Collision.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F4A6; fully - qualified     # 💦
sweat droplets
                            else if (sequence.AsString ==
Emoji.SweatDroplets.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.SweatDroplets.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F4A8; fully - qualified     # 💨
dashing away
                            else if (sequence.AsString ==
Emoji.DashingAway.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.DashingAway.Name;
                                emojiIndexList.Add(eobj);

                            }
```

```
                                // 1F4AB; fully - qualified     # 💫
dizzy
                                else if (sequence.AsString ==
Emoji.Dizzy.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.Dizzy.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                // 1F4AC; fully - qualified     # 💬
speech balloon
                                else if (sequence.AsString ==
Emoji.SpeechBalloon.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.SpeechBalloon.Name;
                                    emojiIndexList.Add(eobj);

                                }
                                //1F5E8 FE0F; fully - qualified     #
🗨 left speech bubble
                                else if (sequence.AsString ==
Emoji.LeftSpeechBubble.ToString())
                                {
                                    EmojiObj eobj = new EmojiObj();
                                    eobj.Index = i;
                                    eobj.EmojiCode = code;
                                    eobj.EmojiName =
Emoji.LeftSpeechBubble.Name;
                                    emojiIndexList.Add(eobj);

                                }
```

```
                            // 1F5E8; non - fully - qualified #
🗨 left speech bubble
                            else if (sequence.AsString ==
Emoji.LeftSpeechBubble.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.LeftSpeechBubble.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F5EF FE0F; fully - qualified
# 🗯 right anger bubble
                            else if (sequence.AsString ==
Emoji.RightAngerBubble.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.RightAngerBubble.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //  1F5EF; non - fully - qualified #
🗯 right anger bubble
                            else if (sequence.AsString ==
Emoji.RightAngerBubble.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.RightAngerBubble.Name;
                                emojiIndexList.Add(eobj);

                            }
```

```
                            //    1F4AD; fully - qualified     #
💭 thought balloon
                            else if (sequence.AsString ==
Emoji.ThoughtBalloon.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName =
Emoji.ThoughtBalloon.Name;
                                emojiIndexList.Add(eobj);

                            }
                            //1F573 FE0F; fully - qualified     #
🕳 hole
                            else if (sequence.AsString ==
Emoji.Hole.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName = Emoji.Hole.Name;
                                emojiIndexList.Add(eobj);

                            }
                            // 1F573; non - fully - qualified #
🕳 hole
                            else if (sequence.AsString ==
Emoji.Hole.ToString())
                            {
                                EmojiObj eobj = new EmojiObj();
                                eobj.Index = i;
                                eobj.EmojiCode = code;
                                eobj.EmojiName = Emoji.Hole.Name;
                                emojiIndexList.Add(eobj);

                            }

                            #endregion
                        }
```

```
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine(ex.Message);

                    }
                }


                string tempStr = string.Empty;
                int tempIndex = 0;
                int newIndex = 0;
                foreach (var item in emojiIndexList)
                {
                    newIndex = tempIndex + item.Index;
                    tempStr = wordData.Remove(newIndex - 1, 2);
                    tempStr = tempStr.Insert(newIndex - 1, " " +
item.EmojiCode + " ");
                    wordData = tempStr;
                    tempIndex = tempIndex +
item.EmojiCode.Length;
                    // Console.WriteLine(wordData);
                }
                //  Console.WriteLine(wordData);

                //Console.WriteLine("Bitdi");

                return wordData.ToLower(); ;
            }
            public static string updateSentence(string wordData)
            {
                string _newWordData = string.Empty;
                wordData = wordData.Trim();
                _newWordData = wordData.Replace(":d"," LLW001 ");
                _newWordData = _newWordData.Replace(":)", "
SHF001 ");
                _newWordData = _newWordData.Replace(":-)", "
SHF011 ");
                _newWordData = _newWordData.Replace(":))", "
SHF002 ");
```

```
                _newWordData = _newWordData.Replace("))", "
SHF003 ");
                return _newWordData;
            }
        }
    }

    namespace PhdTweetApp.BLL.Helper
    {
        public class IOUtil
        {
            public static string GetFunctionRequestID()
            {

                DateTime time = DateTime.Now;
                string guid = time.ToString("yyyy") +
time.ToString("MM") + time.ToString("dd") + time.ToString("HH") +
time.ToString("mm") + time.ToString("ss") + time.ToString("fff");
                System.Threading.Thread.Sleep(1);
                return guid;
            }
        }
    }

    namespace PhdTweetApp.BLL.Helper
    {
        public class TweetUtil
        {

            public static string removeHastagAndOtherTags(string
tweet)
            {

                string newtweet = string.Empty;
                if (!string.IsNullOrEmpty(tweet))
                {
                    string[] array = tweet.Split(' ');
                    for (int i = 0; i < array.Length; i++)
                    {
                        if (getCleanWord(array[i]))
                        {
```

```
                newtweet = newtweet + " " + array[i];

            }
        }


    }

    return newtweet;
}

public static bool getCleanWord(string word)
{
    if (string.IsNullOrEmpty(word))
    {
        return false;
    }
    else
    {
        if (word.Length == 2 && word == "RT")
        {
            return false;
        }
        if (word.Contains('@'))
        {
            return false;
        }
        else if (word.Contains('#'))
        {
            return false;
        }
        else if (word.Contains("http"))
        {
            return false;
        }
        return true;
    }

}
```

```
            public static bool ControlTweet(string ulocation,
bool isretweeted)
            {

                if (ulocation.Contains("zərbaycan") ||
ulocation.Contains("zerbaycan") || ulocation.Contains("akü") ||
ulocation.Contains("aku") || ulocation.Contains("aқy")
                || ulocation.Contains("aki") ||
ulocation.Contains("akı") || ulocation.Contains("zerbaijan") ||
ulocation.Contains("зербайджан")
                || ulocation.Contains("sumqa") ||
ulocation.Contains("sumga")
                || ulocation.Contains("khirdalan") ||
ulocation.Contains("xirdalan")
                || ulocation.Contains("ganja") ||
ulocation.Contains("gəncə"))

                {
                    return true;
                }
                if (isretweeted)
                {
                    return false;
                }
                return false;
            }

            public static int GetWordScore(string tweet)
            {


                int score =
BussinesLogicOperation.getWordScore(tweet);



                return score;
            }
```

```
        }
    }

    namespace PhdTweetApp.DAL.CustomObjects
    {
        public class cstmAllTweets
        {
            public tblAllTweet allTweet { get; set; }
            public List<tblAllTweetHashtag> allTweetHashtag {
get; set; }

        }
    }

    namespace PhdTweetApp.DAL
    {
       public  class OperationLogic
        {

           public bool addCstmAllTweets(cstmAllTweets item) {

               try
               {

                   using (var context = new
PhdTweetAppEntities())
                   {

                       tblAllTweet tweetObj=
context.tblAllTweets.Add(item.allTweet);
                       context.SaveChanges();
                       foreach (var obj in item.allTweetHashtag)
                       {
                           obj.AllTweetID = tweetObj.ID;
                           obj.CreatedAt = tweetObj.CreatedAt;
                           context.tblAllTweetHashtags.Add(obj);
                       }
                       context.SaveChanges();
                       return true;
```

```
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }

            }

            public bool
addtblDictOrthography(List<tblDictOrthography> list) {


                try
                {
                    using (var context = new
PhdTweetAppEntities())
                    {

context.Configuration.AutoDetectChangesEnabled = false;

context.Configuration.ValidateOnSaveEnabled = false;

context.tblDictOrthographies.AddRange(list);
                        context.SaveChanges();
                        return true;
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }


            }
            public List<tblDictOrthography>
gettblDictOrthographies()
            {
```

```
                try
                {
                    using (var context = new
PhdTweetAppEntities())
                    {
                        var items = (from myitem in
context.tblDictOrthographies
                                     select myitem).ToList();
                        return items;
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }

            }

            public bool addtblPozitive(List<tblPozitive> list) {

                try
                {
                    using (var context= new PhdTweetAppEntities())
                    {

context.Configuration.AutoDetectChangesEnabled = false;

context.Configuration.ValidateOnSaveEnabled = false;
                        context.tblPozitives.AddRange(list);
                        context.SaveChanges();
                        return true;
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }

            }
```

```
        public List<tblPozitive> gettblPozitives() {

            try
            {
                using (var context= new PhdTweetAppEntities())
                {
                    var items = (from myitem in
context.tblPozitives
                                 select myitem).ToList();
                    return items;
                }
            }
            catch (Exception ex)
            {

                throw ex;
            }

        }

        public bool addtblNegative(List<tblNegative> list)
        {
            try
            {
                using (var context = new
PhdTweetAppEntities())
                {

context.Configuration.AutoDetectChangesEnabled = false;

context.Configuration.ValidateOnSaveEnabled = false;
                    context.tblNegatives.AddRange(list);
                    context.SaveChanges();
                    return true;
                }
            }
            catch (Exception ex)
            {

                throw ex;
```

```
            }
        }

        public List<tblNegative> gettblNegatives()
        {

            try
            {
                using (var context = new
PhdTweetAppEntities())
                {
                    var items = (from myitem in
context.tblNegatives
                                 select myitem).ToList();
                    return items;
                }
            }
            catch (Exception ex)
            {

                throw ex;
            }

        }

        public List<tblRegion> gettblRegion()
        {

            try
            {
                using (var context = new
PhdTweetAppEntities())
                {
                    var items = (from myitem in
context.tblRegions
                                 select myitem).ToList();
                    return items;
                }
            }
            catch (Exception ex)
            {
```

```
                    throw ex;
                }

            }
            public List<tblRegion>
gettblRegionsByParentCode(string parentCode)
            {

                try
                {
                    using (var context = new
PhdTweetAppEntities())
                    {
                        var items = (from myitem in
context.tblRegions
                                     where
myitem.Parent_Code==parentCode
                                     select myitem).ToList();
                        return items;
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }

            }

            public tblRegion gettblRegionByCode(string Code)
            {

                try
                {
                    using (var context = new
PhdTweetAppEntities())
                    {
                        var items = (from myitem in
context.tblRegions
                                     where myitem.Code == Code
```

```
                                        select
myitem).FirstOrDefault();
                        return items;
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }

            }

            public List<tblEmoji> gettblEmojis()
            {

                try
                {
                    using (var context = new
PhdTweetAppEntities())
                    {
                        var items = (from myitem in
context.tblEmojis
                                     select myitem).ToList();
                        return items;
                    }
                }
                catch (Exception ex)
                {

                    throw ex;
                }

            }
        }
    }


     public class MaximumEntropy
        {
```

```
            //https://docs.microsoft.com/en-
us/dotnet/api/microsoft.ml.textcatalog.featurizetext?view=ml-
dotnet-
1.0.0#Microsoft_ML_TextCatalog_FeaturizeText_Microsoft_ML_Transfor
msCatalog_TextTransforms_System_String_System_String_

            //https://docs.microsoft.com/en-us/dotnet/machine-
learning/resources/glossary
            private static readonly string
BaseDatasetsRelativePath = @"../../../Data";
            private static readonly string DataRelativePath =
$"{BaseDatasetsRelativePath}/tweet_labelledm.txt";

            private static readonly string DataPath =
GetAbsolutePath(DataRelativePath);

            private static readonly string BaseModelsRelativePath
= @"../../../../MLModels";
            private static readonly string ModelRelativePath =
$"{BaseModelsRelativePath}/SentimentModel.zip";

            private static readonly string ModelPath =
GetAbsolutePath(ModelRelativePath);
            public static void RunMaximumEntropy()
            {
                #region try
                // Create MLContext to be shared across the model
creation workflow objects
                // Set a random seed for repeatable/deterministic
results across multiple trainings.
                var mlContext = new MLContext(seed: 1);



                #region step1to3
                // STEP 1: Common data loading configuration
                IDataView dataView =
mlContext.Data.LoadFromTextFile<ModelInputEntopy>(DataPath,
hasHeader: false);
                TrainTestData trainTestData =
mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
```

```
                IDataView trainingData = trainTestData.TrainSet;
                IDataView testData = trainTestData.TestSet;

                // output feature vector depends on these
settings.
                var options = new
TextFeaturizingEstimator.Options()
                {
                    // Also output tokenized words
                    OutputTokensColumnName = "OutputTokens",
                    CaseMode =
TextNormalizingEstimator.CaseMode.Lower,



                    WordFeatureExtractor = new
WordBagEstimator.Options()
                    {
                        NgramLength
                        = 1,
                        UseAllLengths = true,

                    },

                    //CharFeatureExtractor = new
WordBagEstimator.Options()
                    //{
                    //    NgramLength
                    //    = 3,
                    //    UseAllLengths = false
                    //},

                };

                // Define trainer options.
                var entopyoptions = new
SdcaMaximumEntropyMulticlassTrainer.Options
                {
                    // Make the convergence tolerance tighter.
                    ConvergenceTolerance = 0.05f,
```

```
                    // Increase the maximum number of passes over
training data.
                    MaximumNumberOfIterations = 30,
                };


                // STEP 2: Common data process configuration with
pipeline data transformations

                var dataProcessPipeline =
mlContext.Transforms.Conversion.MapValueToKey("Label", "Label")

.Append(mlContext.Transforms.Text.FeaturizeText("Features",
options, nameof(ModelInputEntopy.SentimentText)))

.Append(mlContext.Transforms.NormalizeMinMax("Features",
"Features"))

.AppendCacheCheckpoint(mlContext);


                // STEP 3: Set the training algorithm, then
create and config the modelBuilder
                //var trainer =
mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy(ent
opyoptions);
                var trainer =
mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy(lab
elColumnName: "Label", featureColumnName: "Features")

.Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLa
bel", "PredictedLabel"));
                var trainingPipeline =
dataProcessPipeline.Append(trainer);



                #endregion
```

```
                #region step4
                // STEP 4: Train the model fitting to the DataSet
                ITransformer trainedModel =
trainingPipeline.Fit(trainingData);
                #endregion
                #region step5
                // STEP 5: Evaluate the model and show accuracy
stats
                // Run the model on test data set.
                var transformedTestData =
trainedModel.Transform(testData);
                #endregion

                var metrics = mlContext.MulticlassClassification
                   .Evaluate(transformedTestData);

                PrintMetrics(metrics);

                // STEP 6: Save/persist the trained model to a
.ZIP file
                mlContext.Model.Save(trainedModel,
trainingData.Schema, ModelPath);

                Console.WriteLine("The model is saved to {0}",
ModelPath);



                #endregion

            }

            public static string GetAbsolutePath(string
relativePath)
            {
                FileInfo _dataRoot = new
FileInfo(typeof(Program).Assembly.Location);
                string assemblyFolderPath =
_dataRoot.Directory.FullName;

                string fullPath =
Path.Combine(assemblyFolderPath, relativePath);
```

```
                return fullPath;
            }
            // Pretty-print MulticlassClassificationMetrics
objects.
            public static void
PrintMetrics(MulticlassClassificationMetrics metrics)
            {
                Console.WriteLine($"Micro Accuracy:
{metrics.MicroAccuracy:F2}");
                Console.WriteLine($"Macro Accuracy:
{metrics.MacroAccuracy:F2}");
                Console.WriteLine($"Log Loss:
{metrics.LogLoss:F2}");
                Console.WriteLine(
                    $"Log Loss Reduction:
{metrics.LogLossReduction:F2}\n");


Console.WriteLine(metrics.ConfusionMatrix.GetFormattedConfusionTab
le());
            }
        }

      public class Naive_Bayes
        {
            private static readonly string
BaseDatasetsRelativePath = @"../../../Data";
            private static readonly string DataRelativePath =
$"{BaseDatasetsRelativePath}/tweet_labelled.txt";

            private static readonly string DataPath =
GetAbsolutePath(DataRelativePath);

            private static readonly string BaseModelsRelativePath
= @"../../../../MLModels";
            private static readonly string ModelRelativePath =
$"{BaseModelsRelativePath}/SentimentModel.zip";

            private static readonly string ModelPath =
GetAbsolutePath(ModelRelativePath);
```

```
            public static void RunNaive_Bayes()
            {
                #region try
                // Create MLContext to be shared across the model
creation workflow objects
                // Set a random seed for repeatable/deterministic
results across multiple trainings.
                var mlContext = new MLContext(seed: 1);



                #region step1to3
                // STEP 1: Common data loading configuration
                IDataView dataView =
mlContext.Data.LoadFromTextFile<ModelInput>(DataPath, hasHeader:
false);
                TrainTestData trainTestData =
mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2);
                IDataView trainingData = trainTestData.TrainSet;
                IDataView testData = trainTestData.TestSet;

                // output feature vector depends on these
settings.
                var options = new
TextFeaturizingEstimator.Options()
                {
                    // Also output tokenized words
                    OutputTokensColumnName = "OutputTokens",
                    CaseMode =
TextNormalizingEstimator.CaseMode.Lower,
                    // Use ML.NET's built-in stop word remover
                    StopWordsRemoverOptions = new
StopWordsRemovingEstimator.Options()
                    {
                        Language =
TextFeaturizingEstimator.Language.English
                    },

                    WordFeatureExtractor = new
WordBagEstimator.Options()
```

```
                    {
                        NgramLength
                        = 2,
                        UseAllLengths = true
                    },

                    CharFeatureExtractor = new
WordBagEstimator.Options()
                    {
                        NgramLength
                        = 3,
                        UseAllLengths = false
                    },
                };


                // STEP 2: Common data process configuration with
pipeline data transformations
                // Data process configuration with pipeline data
transformations
                var dataProcessPipeline =
mlContext.Transforms.Conversion.MapValueToKey("Sentiment",
"Sentiment")

.Append(mlContext.Transforms.Text.FeaturizeText("SentimentText_tf"
, "SentimentText"))

.Append(mlContext.Transforms.CopyColumns("Features",
"SentimentText_tf"))

.Append(mlContext.Transforms.NormalizeMinMax("Features",
"Features"))

.AppendCacheCheckpoint(mlContext);


                // STEP 3: Set the training algorithm, then
create and config the modelBuilder
```

```
                var trainer =
mlContext.MulticlassClassification.Trainers.NaiveBayes(labelColumn
Name: "Sentiment", featureColumnName: "Features")

.Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLa
bel", "PredictedLabel"));
                var trainingPipeline =
dataProcessPipeline.Append(trainer);
                #endregion

                #region step4
                // STEP 4: Train the model fitting to the DataSet
                ITransformer trainedModel =
trainingPipeline.Fit(trainingData);
                #endregion

                #region step5
                // STEP 5: Evaluate the model and show accuracy
stats
                var predictions =
trainedModel.Transform(testData);
                // var metrics =
mlContext.BinaryClassification.Evaluate(data: predictions,
labelColumnName: "Label", scoreColumnName: "Score");


                Console.WriteLine("=============== Cross-
validating to get model's accuracy metrics ===============");
                var metrics =
mlContext.MulticlassClassification.Evaluate(data: predictions,
scoreColumnName: "Score", labelColumnName:
"Sentiment",predictedLabelColumnName: "PredictedLabel");
                //
PrintMulticlassClassificationFoldsAverageMetrics(crossValidationRe
sults);
                #endregion


//ConsoleHelper.PrintBinaryClassificationMetrics(trainer.ToString(
), metrics);
```

```
ConsoleHelper.PrintMultiClassClassificationMetrics(trainer.ToStrin
g(), metrics);
                // STEP 6: Save/persist the trained model to a
.ZIP file
                mlContext.Model.Save(trainedModel,
trainingData.Schema, ModelPath);

                Console.WriteLine("The model is saved to {0}",
ModelPath);

                // TRY IT: Make a single test prediction loding
the model from .ZIP file
                ModelInput sampleStatement = new ModelInput {
SentimentText = "ölsem meni pochinkide basdırın 1f611 " };

                #region consume
                // Create prediction engine related to the loaded
trained model
                //var predEngine =
mlContext.Model.CreatePredictionEngine<SentimentIssue,
SentimentPrediction>(trainedModel);
                ITransformer mlModel =
mlContext.Model.Load(ModelPath, out var modelInputSchema);
                var predEngine =
mlContext.Model.CreatePredictionEngine<ModelInput,
ModelOutput>(mlModel);

                // Use model to make prediction on input data
                ModelOutput predictionResult =
predEngine.Predict(sampleStatement);

                //var predEngine =
mlContext.Model.CreatePredictionEngine<ModelInput,
ModelOutput>(trainedModel);

                // Score
                // var resultprediction =
predEngine.Predict(sampleStatement);
                #endregion
```

```
                Console.WriteLine("Using model to make single
prediction -- Comparing actual Sentiment with predicted Sentiment
from sample data...\n\n");
                Console.WriteLine($"SentimentText:
{sampleStatement.SentimentText}");
                Console.WriteLine($"\n\nActual Sentiment:
{sampleStatement.Sentiment} \nPredicted Sentiment value
{predictionResult.Prediction} \nPredicted Sentiment scores:
[{String.Join(",", predictionResult.Score)}]\n\n");
                Console.WriteLine("=============== End of
process, hit any key to finish ===============");
                Console.ReadKey();


                #endregion

            }
            public static string GetAbsolutePath(string
relativePath)
            {
                FileInfo _dataRoot = new
FileInfo(typeof(Program).Assembly.Location);
                string assemblyFolderPath =
_dataRoot.Directory.FullName;

                string fullPath =
Path.Combine(assemblyFolderPath, relativePath);

                return fullPath;
            }
        }

        public class DbOperation
        {
            public List<TweetData> ReadDataFromSQLForBinary()
            {
                string ConnStr = DBUtil.GetConnectionString();
                List<TweetData> tweetDatas = new
List<TweetData>();
                using (SqlConnection con = new
SqlConnection(ConnStr))
```

```
                {
                    con.Open();
                    try
                    {
                        string query = @"select
taz.FullText,taz.CleanText, taz.ConvertedText,taz.Cost,
                                                CASE WHEN
taz.cost = 0 THEN 'Neutral'

                                                    WHEN
taz.cost > 0 THEN 'Positive'

                                                    WHEN
taz.cost < 0  THEN 'Negative'

                                                    END AS
Sentiment ,
                                                CASE
                                                    WHEN
taz.cost>0 THEN  1
                                                   WHEN
taz.cost<0  THEN 0
                                                        END AS
SentimentReal ,
                                            taz.Subject
from[dbo].[tblAllTweetAZ] taz
                                            where taz.Cost!=100 and
taz.Cost!=-100  and taz.Cost is not null;";
                        using (SqlCommand command = new
SqlCommand(query, con))
                        {

                            var reader = command.ExecuteReader();
                            while (reader.Read())
                            {
                                tweetDatas.Add(new TweetData()
                                {
                                    FullText =
reader.GetStringOrEmpty(0),
                                    CleanText =
reader.GetStringOrEmpty(1),
```

```
                                    ConvertedText =
reader.GetStringOrEmpty(2).Trim().Replace(System.Environment.NewLi
ne, " "),
                                    Cost =
reader.GetDecimalOrDefaultValue(3),
                                    Sentiment =
reader.GetStringOrEmpty(4),
                                    SentimentReal =
reader.GetInt32OrDefaultValue(5),
                                    Subject =
reader.GetStringOrEmpty(6),
                                });
                            }
                        }

                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine("Something went wrong "
+ ex.Message);
                    }
                    return tweetDatas;
                }

                Console.Read();
            }
            public  List<TweetData> ReadDataFromSQLMulti()
            {
                string ConnStr = DBUtil.GetConnectionString();
                List<TweetData> tweetDatas = new
List<TweetData>();
                using (SqlConnection con = new
SqlConnection(ConnStr))
                {
                    con.Open();
                    try
                    {
                        string query = @"select
taz.FullText,taz.CleanText, taz.ConvertedText,taz.Cost,
                                                CASE WHEN
taz.cost = 0 THEN 'Neutral'
```

```
                                                                WHEN
taz.cost > 0 THEN 'Positive'

                                                                WHEN
taz.cost < 0  THEN 'Negative'
                                                                END AS
Sentiment ,
                                                        CASE
                                                                WHEN
taz.cost=0  THEN 0
                                                                WHEN
taz.cost>0 THEN  1
                                                                WHEN
taz.cost<0  THEN 2
                                                        END AS
SentimentReal ,
                                                taz.Subject
from[dbo].[tblAllTweetAZ] taz
                                                where taz.Cost!=100 and
taz.Cost!=-100  and taz.Cost is not null;";
                        using (SqlCommand command = new
SqlCommand(query, con))
                        {

                            var reader = command.ExecuteReader();
                            while (reader.Read())
                            {
                                tweetDatas.Add(new TweetData()
                                {
                                    FullText =
reader.GetStringOrEmpty(0),
                                    CleanText =
reader.GetStringOrEmpty(1),
                                    ConvertedText =
reader.GetStringOrEmpty(2).Trim().Replace(System.Environment.NewLi
ne, " "),
                                    Cost =
reader.GetDecimalOrDefaultValue(3),
                                    Sentiment =
reader.GetStringOrEmpty(4),
```

```
                                    SentimentReal =
reader.GetInt32OrDefaultValue(5),
                                    Subject =
reader.GetStringOrEmpty(6),
                                });
                            }
                        }

                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine("Something went wrong
"+ex.Message);
                    }
                    return tweetDatas;
                }

                Console.Read();
            }
        }
```